IDT77211 NICStAR™

User Manual

Version 1.0
Released Edition
February 26, 1997

Find IDT on the World Wide Web at: www.idt.com
or Call us at 1-800-345-7015
IDT is located at:
2975 Stender Way
Santa Clara, California, USA 95054-3090

0.0 Introduction to the User's Manual Document

0.1 Intended Audience

The intended audience of this document is programmers who will write device drivers and/or higher layer networking application software using the NICStAR in their systems. Also, system/hardware engineers who want to evaluate the NICStAR for their system requirements can read the Theory of Operation section.

0.2 How to use this manual

This document is divided into the following sections:

Section 1: NICStAR Overview

Describes the overview and key features of the NICStAR.

Section 2: Theory of Operation

Describes the functional components of the NICStAR and how they inter-operate with the host system.

Section 3: Data Structure and Registers

Describes the required data structures associated with the NICStAR and all registers inside the NICStAR.

Section 4: ATM Layer Cell Handling

Describes how ATM received cells are being handled by the NICStAR.

Section 5: Recommended NICStAR Service Algorithms

Provides recommended algorithm for servicing the NICStAR. These examples keep in mind performance issue.

Section 6: Glossary

Explains terms used in this document.

Section 7: Appendix

This section provides examples and application notes on how to use the NICStAR.

0.3 Related documents

NICStAR IDT77211 Data Sheet:

This document provides additional information on the NICStAR. It contains pinouts and all the AC/DC electrical parameters of the device.

0.4 Revision History

2/26/97 Created Frame Maker Document from the 77201 User's Manual

Table of Contents

1 NICStAR Overview

The IDT77211 NICStAR™ PCI ATM Controller is a member of IDT's family of products for Asynchronous Transfer Mode (ATM) networks. The NICStAR performs both the ATM Adaptation Layer (AAL) Segmentation and Reassembly (SAR) function and the ATM layer protocol functions.

The NICStAR's architecture buffers Convergence Sublayer Protocol Data Unit (CS-PDU) in host memory rather than in on-board local memory to lower implementation costs. It uses the PCI bus master capability to access the host memory to reduce host CPU utilization. The NICStAR controls all AAL SAR and ATM layer run-time operations and performs all key data transfers between the host system and itself as a PCI bus master; the device driver provides the NICStAR with buffer and control descriptors during run-time.

When receiving cells, the NICStAR performs a table lookup using the connection number, in the cell header, to determine where to place the cell payload in the host memory. The ATM cell payloads received from the network are then placed directly into host memory to form CS-PDUs.

When transmitting cells the CS-PDUs are queued in host memory by the host driver software. As the NICStAR segments CS-PDU buffers into ATM cell payloads, it appends the respective cell headers to create 53-byte ATM cells. The HEC byte is inserted as a place holder by the NICStAR. The cells are then transmitted to the PHY device as allocated by a scheduling table. The PHY device then calculates and replaces the Header Error Check (HEC) byte to form the complete 53-byte ATM cell.

1.1 NICStAR's Key features

- Provides sustained full-duplex Segmentation and Reassembly (SAR) up to and including 155 Mbps wire-speed.
- Stand alone SAR function, no attached processor required.
- Performs the ATM layer protocol functions.
- Uses host memory for both segmenting and reassembling CS-PDUs.
- Supports Constant Bit Rate (CBR), Variable Bit Rate (VBR) and Unspecified Bit Rate (UBR) service classes and Operations And Maintenance (OAM) cells.
- Supports AAL5 and AAL3/4 ATM cell formats as well as “AAL0” and “Raw Cell” formats.
- NICStAR provides a streamlined hardware interface to reduce board cost and enhance system performance:
  - Direct interface to 32-bit/33 MHz PCI bus.
  - Supports Big and Little endian.
  - Performs all key data transfers as a PCI bus master.
  - Direct interface to UTOPIA-compliant PHY-TC components.
  - Requires only a small amount of local SRAM: 32K x 32 or 128K x 32, uses 15ns SRAM when operating at 50MHz (SAR_CLK).

- Flexible transmission architecture:
  - Supports any buffer byte alignment condition for AAL5 and “AAL0” formats.
  - Supports 16,000,000 simultaneously open transmit connections.
  - Includes a 9-cell Tx FIFO on-chip.
  - Transmission schedule table provides precise control of cell timing.

- Flexible receive architecture:
  - Supports 16K simultaneously open receive connections.
  - 315-cell Rx FIFO in local SRAM.
  - Two receive buffer pools available for independant use or chained together.

2 NICStAR Theory of Operation

2.1 Introduction

The IDT77211 NICStAR™ PCI ATM Controller is a member of IDT's family of products for Asynchronous Transfer Mode (ATM) networks. The NICStAR performs the ATM Adaptation Layer (AAL) Segmentation and Reassembly (SAR) function. It is intended for use in Desktop/Workstation Network Interface Card (NIC) products as well as Switch/Router configurations where SAR functionallity is needed. For this reason the NICStAR is designed for low cost. The NICStAR is optimized to take advantage of large amounts of memory available local to the Host CPU.

Throughout this manual, the terms Head Pointer and Tail Pointer will be used. To avoid confusion, please read Head Pointer as a pointer where data will be read from and Tail Pointer as a pointer where data will be written to a queue.

2.1.1 Segmentation and Reassembly Location

The location of the Convergence Sublayer-Protocol Data Unit (CS-PDU) buffer which is being segmented or reassembled has a large impact on system cost and performance. There are two major NIC architectures for segmentation and reassembly storage: either local to the SAR or local to the host CPU.

Storing CS-PDUs local to the SAR is used by systems that have a bus with less sustained bandwidth than the ATM connection. CS-PDUs are transferred at a low rate and then later burst across the network at line rate. The PDUs must be copied twice which adds to latency of network protocols. Since the PDUs are kept in local memory on the NIC, there is a physical limitation (without using a complex algorithm) on the number of PDUs that can be handled simultaneously.

Storing CS-PDUs local to the host CPU is used in systems which use a high performance bus between the SAR and host CPU local memory. The data which is to be segmented/reassembled can be kept local to the CPU. This removes the constraint of fixed memory. It reduces latency of doing an extra copy from the network to the SAR's local memory and again to the host's local memory.

Since NICStAR is designed to be in a system where large host CPU local DRAM memory and a high speed bus is available, all of the segmentation and reassembly is performed with the CPU local memory model. The NICStAR thus allows for maximum flexibility in buffer sizes and volume of PDUs handled at a time.

2.1.2 Data Movement

In a Desktop or Workstation environment utilizing a bus like PCI, accesses between the CPU and local DRAM are much faster than accesses between the CPU and a peripheral. With this in mind, the NICStAR has been designed to move all data and large control structures in PCI Bus Master block mode whenever possible. The CPU is used during data transfer setup and is only involved in small data movement. As an example, cell payloads are 12 words, which are always moved by the NICStAR in PCI bus master fashion. In this

way both the host CPU and PCI bus overhead are minimized.

2.1.3 Support for ATM Service Categories

The NICStAR provides for two basic groups of channels for transmission of cells. Each group has a different mechanism for controlling the cell transmission rate: Fixed and Variable. The Fixed Cell Rate group of channels are controlled by a schedule table. The Variable Cell Rate group of channels are controlled by rate counters for connections that expect to vary their transmission rate "On-the-fly" on a buffer by buffer basis. The driver can then implement ATM service categories by placing cells and buffers to be transmitted in the appropriate queues.

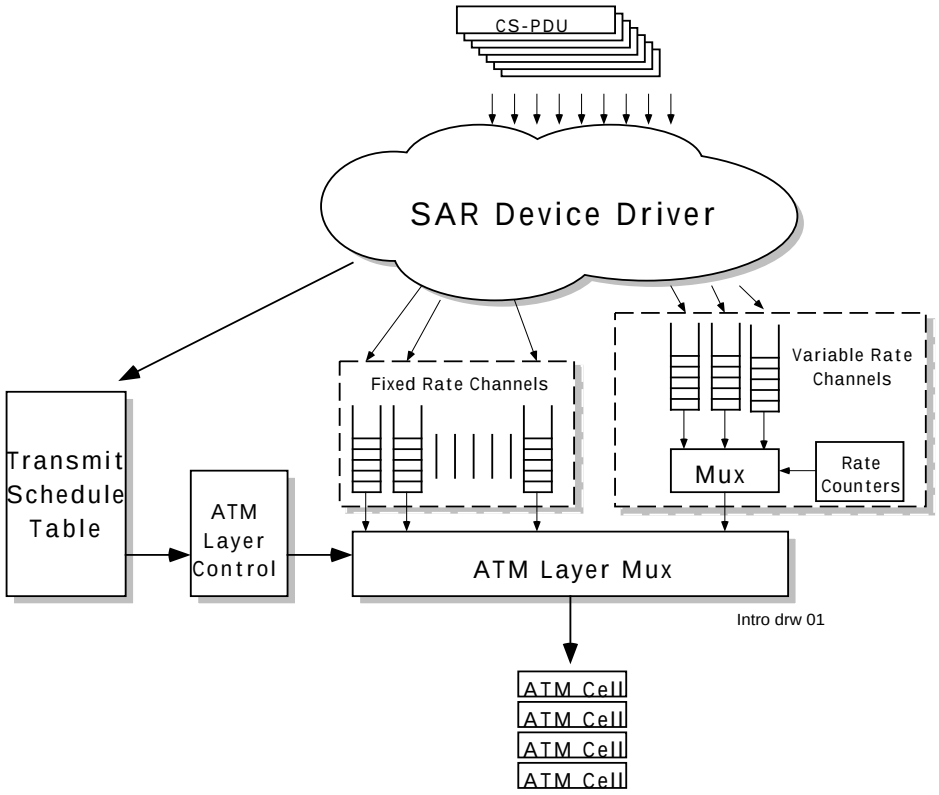


Figure 1: ATM Services supported by the NICStAR

NICStAR has been designed to handle both fixed rate and variable rate transmission where timing of cells is most important. To this end, NICStAR employs a ***Transmit Schedule Table*** for precise control of cell transmission timing for fixed rate channels. The fixed rate channels are therefore suitable for Constant Bit Rate (CBR) and Variable Bit Rate (VBR) service classes where Cell Delay Variation is to be kept very small.

All remaining bandwidth is allotted to variable rate traffic such as Unspecified Bit Rate (UBR) and Available Bit Rate (ABR) service classes. The variable rate traffic is throttled by rate timers which are used for flow control.

Moderate response time ABR service can be implemented by using a combination of driver, table driven computation, precomputed traffic shapes and utilizing transmit small buffers/queues.

2.2 The Network Interface Card with NICStAR

The Network Interface Card using NICStAR is a highly integrated solution which consists of NICStAR connecting directly to the PCI bus, an UTOPIA bus for a PHY device, and local SRAM for key data structures like a Connection Table and Transmit Schedule Table. Optional devices can be attached like PROM and EEPROM as the application dictates. When creating a NIC solution with NICStAR there is only one PCI bus load.

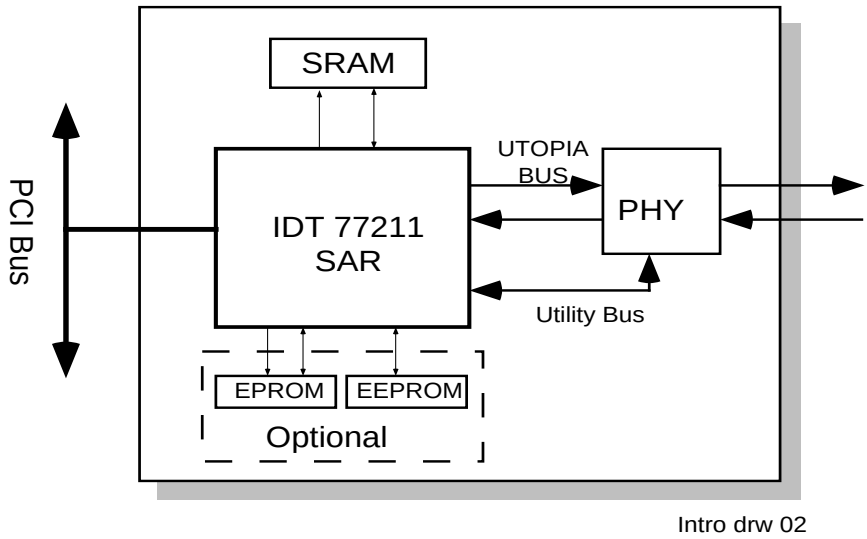


Figure 2: Network Interface Card using NICStAR

There are two configurations of SRAM that can be connected to the NICStAR. The 32K word configuration (four 32K x 8-bit devices) provides a connection table of 4K Tx connections and 16K Rx connections and the 128K word configuration (four 128K x 8-bit devices) provides 4K Tx connections and 64K Rx possible connections.

The PCI interface is clocked from the PCI bus, while the state machine operation of the NICStAR is clocked by a 50 MHz oscillator for 155 Mbps operation and 32MHz oscillator for 25Mbps operations. The PHY bus (Utopia I compliant) can be clocked by the NICStAR clock divided by two (output from the NICStAR) or an optional oscillator.

2.3 Intrinsic Elements of NICStAR

2.3.1 Registers

The NICStAR has two major groups of registers: PCI configuration and NIC operation. The PCI configuration registers are used by the PCI BIOS to perform operations like mapping the NICStAR in to memory and I/O spaces. The NICStAR has NIC specific registers which are used to address the resources of the NIC.

The NIC specific registers include a set of data and a command register to access the SRAM and Utility bus. Also included are configuration and status registers for the NICStAR device, as well as base and pointer registers for the status queues. The general register can be used to access the EEPROM.

2.3.2 SRAM

The local SRAM attached to the NICStAR is used to hold data structures that describe information such as Receive Connection Table, Free Buffer Queues, Segmentation Channels, and Transmit Schedule Table. The SRAM can be accessed via the Command and Data Registers in the NICStAR's register set. Data structures, such as the ***Free Buffer Queue*** and the ***Receive Connection Table,*** have their own special commands.

2.3.3 Utility Bus

The Utility Bus is used to communicate with the Framer and Physical Media devices. It is an 8-bit multiplexed address and data bus with read/write strobes. There are two Chip Selects (CS) which can be used to address in simple fashion up to two devices. The registers on PHY devices can be accessed via the ***Command*** and ***Data*** Registers in the NICStAR's register set.

2.3.4 EPROM

An optional EPROM can be connected to the NICStAR which can be used to store a network boot time driver. The EPROM can be accessed via the PCI bus interface of the NICStAR by setting up the ***ROM Expansion Base Address*** register which is one of the ***PCI Configuration Registers***. Once set, the EPROM is mapped into memory space.

2.3.5 EEPROM

The NICStAR has four pins (three outputs, one input) which can be used to access an optional EEPROM. The four pins are parallel I/O pins and can be used in a serial fashion to connect to an EEPROM. The pins are accessed via the General Purpose register of the NICStAR. The host driver software must toggle the pins in an appropriate fashion to access specific EEPROM devices.

2.3.6 Use of Host DRAM

The NICStAR uses Host DRAM to optimize system performance. On transmit containing data to be sent or identified by the Host Driver and described in a Transmit Segmentation Channel Queue (SCQ). The SAR DMA's the information from the SCQ as well as the date itself leaving the Host free for other activities. Upon successful transfer, the SAR reports to the Host via a

Transmit Status Queue also in Host DRAM.

When receiving cells, the Host Driver has identified memory locations (Free Buffers) in which to receive data. The SAR DMA's the reassembled cells to Host memory's Free Buffers and notifies the Host via the Receive Status Queue upon completion.

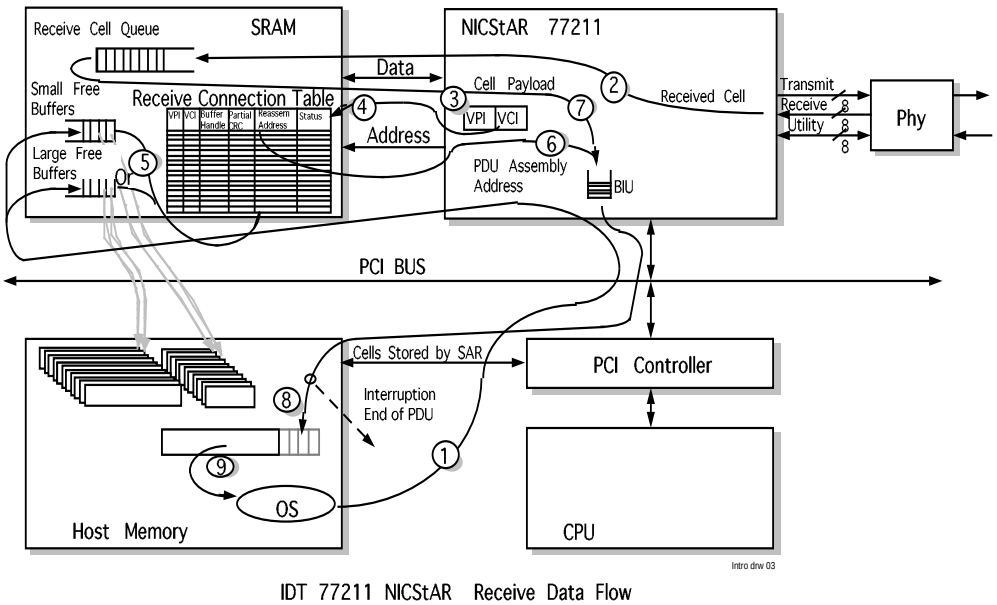
2.4 Overview of Data Flow

This section will describe the data flow of the NICStAR and its associated components in the host system.

2.4.1 Receive Data Flow

Cells are received by NICStAR on the UTOPIA bus and placed into the SRAM connected to NICStAR. The cells are then divided into the header and the payload. From the header, the Virtual Connection number (VPI/VCI) is used to index a connection table, which is used to select a unique buffer to receive the cell payload. A free pool of small and large buffers is used as a source of new buffers to be drawn as the NICStAR reassembles cell payloads into CS-PDUs.

The receive data flow diagram of the NICStAR is shown below along with the description of the key steps as designated by the numbering circles.



IDT 77211 NICStAR Receive Data Flow

Figure 3: NICStAR Receive Data Flow

(1). Before reassembly may begin, the device driver must provide the NICStAR with a supply of host memory locations (buffers) which may be used for

reassembly of ATM cell payloads into CS-PDUs. The start address of each buffer allocated for reassembly, called Small Free Buffers and Large Free Buffers, must be programmed into the local SRAM's Small Free Buffer Queue and Large Free Buffer Queue, respectively.

(2). A 53-byte ATM cell received from the PHY is immediately written by the NICStAR into the local SRAM's Receive Cell Queue (315 cell FIFO). The memory map for the Rx Cell FIFO is in SRAM memory between 1E800 and 1F7FF. The NICStAR writes the ATM cell header without the HEC byte, since the HEC byte was calculated and compared within the PHY prior to being received by the NICStAR.

(3). The ATM cell header is read by the NICStAR.

(4). The NICStAR uses the VPI/VCI field of the ATM cell header to index into the Receive Connection Table, which contains the following information:

- VPI/VCI (unique for each virtual connection)
- Buffer Handle (virtual start address of a free buffer)
- Partial CRC value (for AAL5 PDU)
- Reassembly Address (from Free Buffer Queues)
- Status (AAL format, etc.)

(5). There are two Free Buffer Queues available for reassembly of received PDUs. These can be used independent of one another or can be chained together (Small and then as many large buffers as needed). Each connection can be configured to reassemble using either method. When the chain method is applied, assuming this is the first ATM cell received for this CS-PDU, the first free buffer address in the Small Free Buffer Queue is copied into the Receive Connection Table entry for the specified virtual channel (VC). As additional cells are received for this CS-PDU, cell payloads are deposited into host memory at remaining addresses pointed to by this Small Free Buffer. Once the Small Free Buffer memory area is exhausted, subsequent free buffers (as needed) are copied from the Large Free Buffer Queue to finish reassembly of the PDU.

When the independent buffer pools are used, the first free buffer address will be pulled from the free buffer queue as specified in the connection table.

When Large Free Buffers are used, and the PDU cannot be contained in the Large Free Buffer, a second Large Free Buffer is selected and chained automatically.

There is a limitation with the Small Free Buffer Queue when using the buffer queues as independent pools. If only Small Free Buffers are to be used and the PDU cannot be contained within the Small Free Buffer, the next free buffer chained will be a Large Free Buffer. Since Small Free Buffers can be configured up to 2K bytes in size in the 77211, no practical complications should arise from this limitation.

(6). The NICStAR writes the start address for the Small Free Buffer to its Bus Interface Unit (BIU).

(7). The NICStAR writes the 12 word ATM cell payload to its BIU.

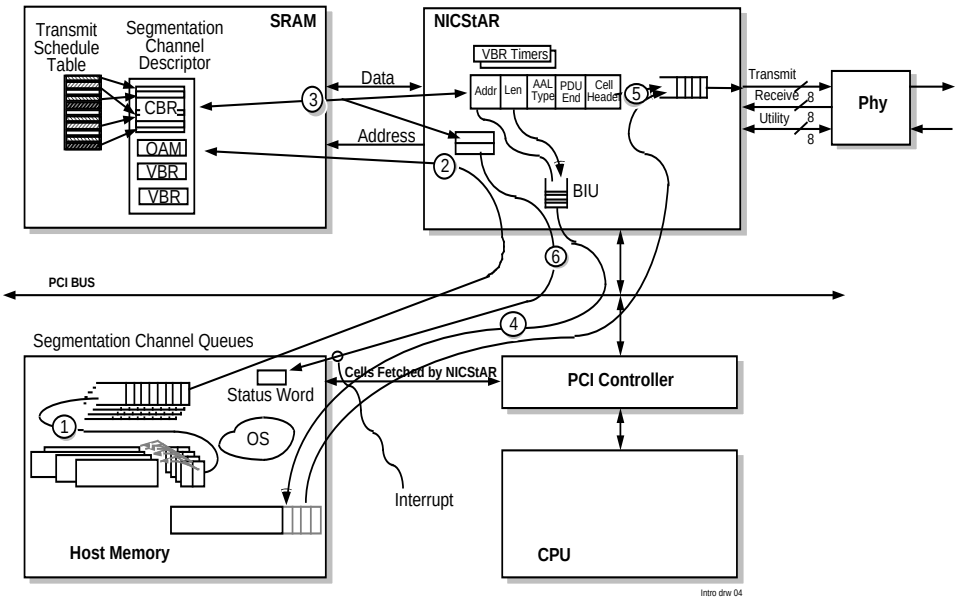
(8). The NICStAR performs a PCI Bus Master transfer of the 48-byte ATM cell payload to the specified Free Buffer in host memory. After completely filling any Small or Large Free Buffer in host memory, the NICStAR writes the start address of the buffer to the Receive Status Queue, located in host memory. As additional Large Free Buffers are filled with ATM cell payloads, the NICStAR writes the start addresses of the Large Free Buffers to the Receive Status Queue for the specified VC. After the NICStAR detects an end of PDU, it may (optionally) generate an interrupt, informing the host system to service the Receive Status Queue.

(9). After an "end of PDU" is detected, the device driver reads the Receive Status Queue, generates a list of host memory buffer addresses which constitute the received CS-PDU, and then provides the list of addresses to the application program(s) for converting back to user data.

2.4.2 Transmit Data Flow

When transmitting cells the SAR fetches cell payloads from host memory buffers as designated by buffer descriptors kept in host memory in transmit ready queues. The timing of segmenting cells from the host memory buffers is controlled first by a schedule table and secondarily by a rate counters. As cell payloads are fetched and formed into complete ATM cells they are stored in an on chip cell queue to provide a buffer against time when the PCI bus may not become available on a timely basis.

The transmit data flow diagram of the NICStAR is shown below along with the description of the key steps as designated by the numbering circles.



IDT 77201 NICStAR Transmit Data Flow

Figure 4: NICStAR Transmit Data Flow

Explanation of Figure 4:

1. As a CS-PDU becomes available for transmit, the device driver creates Transmit Buffer Descriptors (TBDs) for the sequence of buffers in host memory which constitute the CS-PDU, and then writes the TBDs into a TBD queue, located in host memory.

2. The device driver then causes the NICStAR to copy the first one or two TBDs to local SRAM.

3. The NICStAR reads the first TBD. The ATM cell header, also part of this TBD, is loaded into the output FIFO. During this process, a HEC byte place holder (00h) is added as the fifth byte of the header.

4. The PCI bus is arbitrated using the address and length taken from the TBD.

5. The ATM cell payload is transferred from host memory to the output FIFO using PCI Bus Master mode. On completion, the 53-byte ATM cell is transferred out of the NICStAR via the UTOPIA interface.

6. Status information is returned to the host system to communicate transmission state, error conditions, etc.

2.5 Detailed Description of Reassembling ATM Cells

Receiving cells and reassembling CS-PDU is controlled by the OS driver by providing pointers to free buffers to the NICStAR and processing filled data buffers.

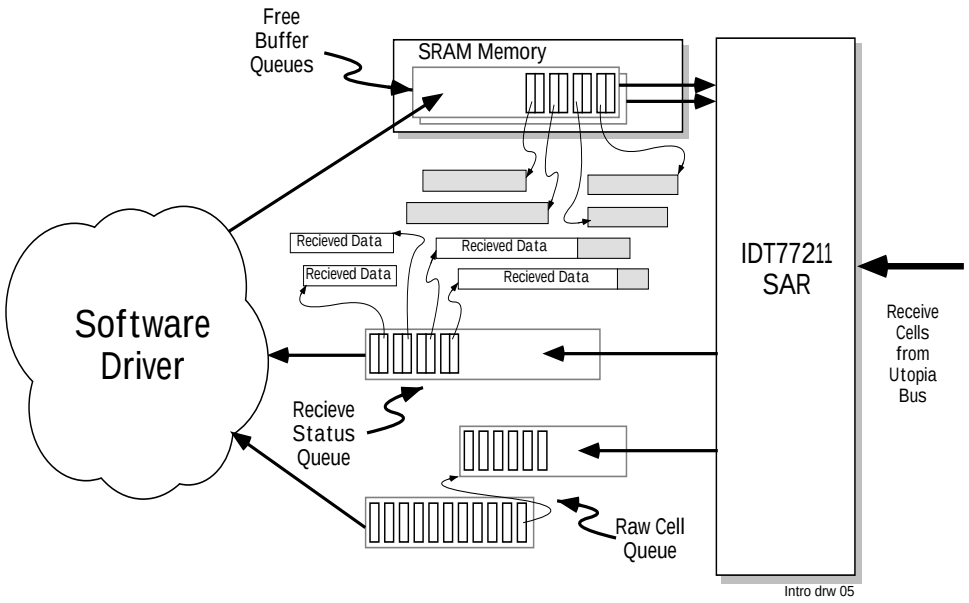


Figure 5: Receive Buffer Overview

Free Buffers are loaded onto ***Free Buffer Queues*** which are located in the NICStAR's local SRAM. Cell payloads are then fetched from the local SRAM and then placed into buffers in the host memory as determined by the connection table. Receive status (pointers to completed buffers) is returned in the ***Receive Status Queue*** maintained in CPU's local DRAM. CS-PDUs can be made up of multiple buffers and chained together via the ***Receive Status Queue***.

In the work station environment it is assumed the protocols like Internet Protocol will be transmitted often. It is known that a high percentage of all CS-PDUs transmitted will be short PDUs holding acknowledgment type information. For this reason, NICStAR handles two different size PDU buffers: large and small. Each new CS-PDU fills a small free buffer and, subsequently, large buffers.

A ***Connection Table*** is the primary control structure for keeping track of which connection is open and the state of the connection. The ***Connection Table*** is kept in the local NICStAR SRAM.

2.5.1 Receive Queues

There are three queues which are used for receiving cells and CS-PDUs: ***Free Buffers***, ***Receive Status***, and ***Raw Cells***. In addition to these three queues, the

NICStAR maintains a receive FIFO in local SRAM. This FIFO holds up to 315 cells and is used as a buffer against PCI bus latency.

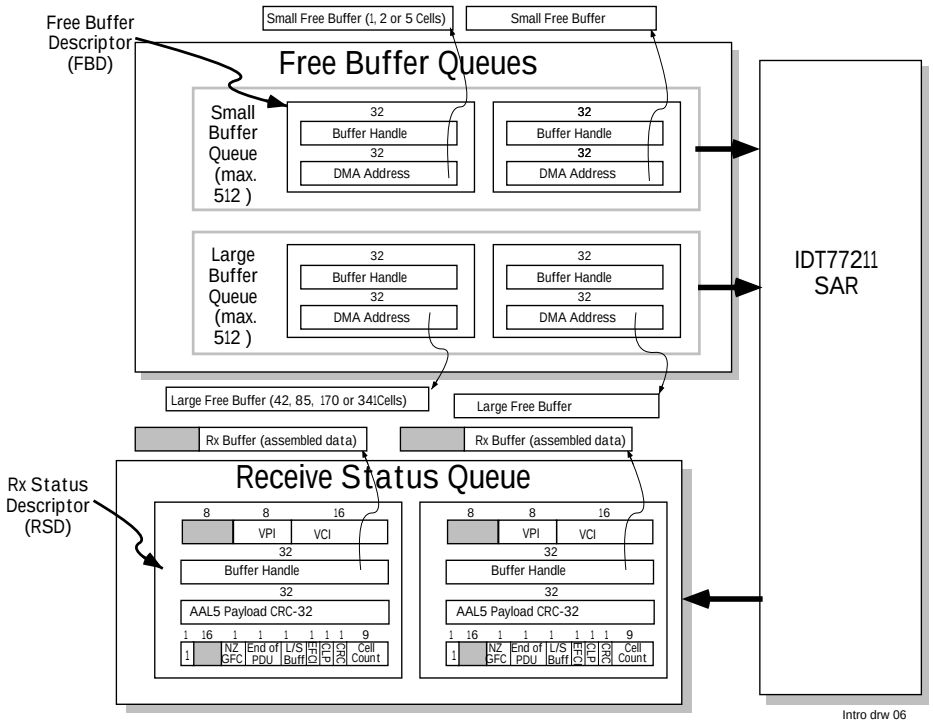


Figure 6: NICStAR Receive Buffer Queue Processing

2.5.1.1 Free Buffers

There are two free buffer queues: ***Small Free Buffers*** and ***Large Free Buffers***. The free buffer queues are kept in the NICStAR's local SRAM. Up to 512 free buffers can be loaded onto each of these queues at any one time. Free buffer queue entries are queued by the driver and de-queued by the NICStAR as needed.

The size of the ***Small*** and ***Large Buffers*** is determined by values programmed by the driver into the ***Configuration Register*** (Register #5). The size of small free buffers can be 1, 2, 5 or 42 cells (48, 96, 240, 2K bytes). The size of large free buffers can be 42, 85, 170,or 341 cells (2K, 4K, 8K, or 16K bytes). The size can only be set at system initialization. Free buffers must be word (modulo 4 bytes) aligned.

A Free buffer is described by two 32-bit words, a DMA Address and a ***Buffer Handle***. The ***DMA Address*** is the physical address of the buffer in the CPU's local DRAM. The ***Buffer Handle*** is for use strictly by the OS software driver. It is a 32-bit number which the NICStAR does not change. A driver could use the ***Buffer Handle*** as a Virtual Address, an index into a table, a field of predefined bits, etc.

2.5.1.2 Receive Status

The ***Receive Status Queue***, located in the host memory, is where the NICStAR writes pointers to buffers which have been filled as a result of receiving data. When a buffer is completely filled or an End Of PDU is sensed, the NICStAR places a four word descriptor into the ***Receive Status Queue***. The four word entry contains the ***Virtual Connection*** number, the ***Buffer Handle***, the ***AAL5 CRC-32*** (valid only if an AAL5 connection), and a ***Receive Status*** word. The status word indicates information about the buffer such as Explicit Forward Congestion Indication (EFCI) status, buffer size, end of PDU, and if there was a CRC error for AAL5.

The ***Receive Status Queue*** is written by the NICStAR and read (consumed) by the OS device driver. ***Receive Status Queue Entries*** have been designed such that the ***Receive Status Queue*** can be processed very efficiently by reading only the CPU's local memory. Each entry of the status queue contains a valid bit which allows the driver to monitor NICStAR progress without reading from the NICStAR. Traditional methods of polling the tail and head pointers located in the NICStAR can use up many valuable cycles on the PCI bus.

The size of the ***Receive Status Queue*** is set in the ***Configuration*** register. The range is 2K (default) to 8K bytes (128 to 512 entries). The device driver loads the address of the base of the ***Receive Status Queue*** in the ***Receive Status Queue Base*** register (register #7). This base address must be a multiple of the size of the queue, because the LSBs (Least Significant Bit) of the register are forced to 0. See the ***Receive Status Queue Base*** register (register #7) for more information.

2.5.1.3 Raw Cell

The ***Raw Cell Queue*** consists of one or more ***Large Free Buffers*** in host memory. It is built and used by the NICStAR for storing complete ATM cells which should not or could not be reassembled. Examples of such cells are flow control Resource Management (RM) cells, Operations and Maintenance (OAM) cells, or cells for which the AAL type field is set to "non-AAL" in the ***Receive Connection Table***. In these cases, the NICStAR will transfer 52-bytes of the ATM cell directly to the Raw Cell Queue. The queue will contain 4-bytes of the cell header (the HEC byte is not transferred), 3 words of padding (zeros), and the 12 word payload.

The NICStAR chains buffers together to form the free buffer queue. When the last cell slot in a buffer is going to be written, the NICStAR first pulls another ***Large Free Buffer*** from the queue in local SRAM. Then it writes the ***Buffer Handle*** and ***DMA Address*** for this new ***Large Free Buffer*** into word locations 1 and 2, respectively, of the current buffer's last cell slot (the remaining 14 words are padded with 0 hex values). The NICStAR then performs a standard raw cell transfer to the first cell slot of the new buffer. When processing the ***Raw Cell Queue***, the device driver uses the ***Buffer Handle*** and ***DMA Address*** values to access the next buffer in the chain.

The ***Receive Raw Cell Queue Tail*** pointer register points to the tail of the queue. At reset, the NICStAR reads the first ***Large Free Buffer*** and assigns it to be the first Receive Raw Cell buffer. The driver should initialize its own Raw Cell

Head pointer to the first ***Large Buffe***r that it loaded at initialization.

Cells meeting the following conditions will be deposited in the ***Raw Cell Queue:***

1. Identified as "Raw Cell" in the ***Receive Connection Table's*** AAL Type field (bits 18-16).
2. A cell with an unknown VPI/VCI (optional).
3. A cell's ***PTI*** = 100 or 101 for OAM F5 cell
4. A cell's ***VC*** = F4.

2.5.2 Receive Connection Table

The ***Receive Connection Table*** is located in local SRAM. It is indexed by a Virtual Connection number (VC) derived from the VPI and VCI fields of an incoming cell. The combined VPI (8 bits) and VCI (16 bits) comprise the address. The NICStAR decodes up to 14 of the 24 bits. Each table entry is made up of four words used to control how the NICStAR handles its particular VC. The connection table is set up and maintained by the driver and the NICStAR updates it as to the status of each CS-PDU received on a connection.

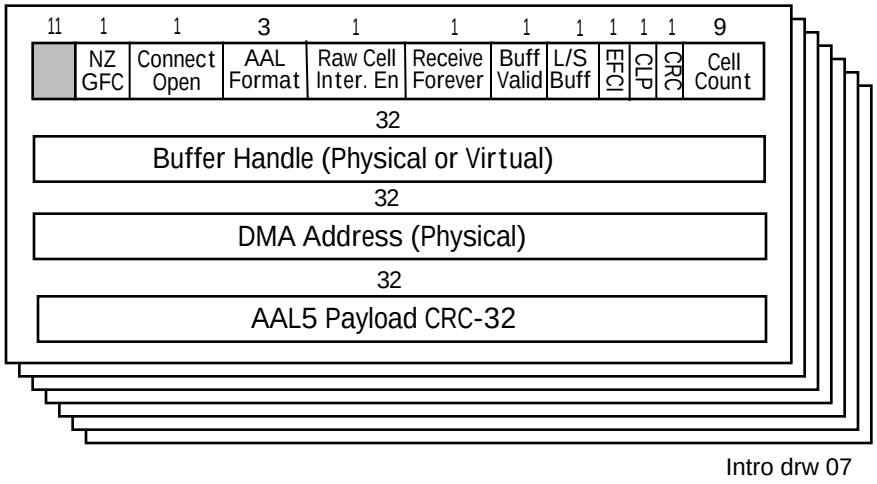


Figure 7: Receive Connection Table

Each connection entry in the ***Receive Connection Table*** contains four words of status which indicate: Control/Status Information, Buffer Handle, DMA Address, Intermediate AAL5 Payload CRC-32.

The number of bits used for each LSB portion is programmable by the device driver at initialization via the ***Configuration Register*** (register #5, bits 18-19). The NICStAR will use from 0 to 8 bits for the VPI field comparison and from 4 to 14 bits for the VCI field comparison. The remaining VPI and VCI bits of the received cell, the MSB (Most Significant Bit) portions, are matched against a mask of the VPI and VCI MSB portions in the VPI/VCI MSB Mask register to ensure complete decode of the VPI and VCI fields. If the mask feature is set

to other than zero, the NICStAR will not be able to handle ATM traffic management cells. Such use of the mask register will preclude use of VC's 0-32, which is the range of VC's used for traffic management cells.

2.5.3 Receive Operations and Algorithms

The NICStAR architecture is optimized with the intent such that the NICStAR moves large block of data in PCI Bus Master mode and the CPU performs less frequent accesses that must be required by the CPU. In most cases, handling of queues can be performed in several ways. The algorithm outlined in this section are suggestions which are intended for optimizing data movement and system performance.

2.5.3.1 Opening a connection

The driver opens or closes a connection by writing an open/close command to the NICStAR via the ***Command*** register (Register #5).

2.5.3.2 Receive Interrupts

The NICStAR has multiple interrupt sources as a result of receive operations. When an interrupt is asserted internally in the NICStAR, an interrupt is posted to the Host CPU via the PCI bus. The driver must then read the ***Status*** register to determine the cause of the interrupt. The driver may disable specific interrupt sources by masking off the appropriate bit in the ***Configuration*** register (register #5).

Often drivers wish to batch the processing of the PDU receive queues. This can be achieved with the NICStAR in one of two ways. The first involves a delayed interrupt. The NICStAR provides an interrupt that indicates ***End-of-PDU*** condition. There is a mechanism to hold off the assertion of the ***End-of-PDU*** interrupt by 314µs, 624µs, or 899µs (programmed via the ***Configuration*** register (Register #5). Use of this feature allows several PDUs to arrive before the interrupt is asserted. The second method uses the same interrupt, but without the delay. The driver responds at a higher priority and posts the actual processing task to a lower priority routine.

Note: Because of the architecture of the NICStAR, there can be a delay between the time a register is updated, and the time the data that caused the update arrives in host memory. There is no delay between the time an interrupt occurs and the time that the data causing the interrupt arrives in host memory. It is necessary to compensate for this timing delay when writing an ISR (Interrupt Service Routine). Example time line:

1) Host receives an interrupt from the NICStAR: service routine is called.

2) Another interrupt causing event occurs in the NICStAR.

3) ISR reads the status register, which now indicates that two interrupts are pending.

4) ISR services the two interrupt conditions and finds that one of them has no data in host memory (timing delay).

5) Service routine ignores the no-data interrupt and exits.

6) Data for second interrupt (from step 2) arrives in host memory and is lost.

To solve this problem an ISR for the NICStAR can be designed to hand off the no-data interrupt to a low-priority routine. This routine will wait for the data to show up in memory. Such a routine could be written using "sleep" and "timer" functions that would avoid the overhead of continuous polling.

2.5.3.2.1 End-Of-PDU received

This interrupt is asserted when the last cell of a PDU is completely passed over the PCI bus. The configuration register can be used to specify an interrupt assertion hold off 0µs, 314µs, 624µs, or 899µs. This interrupt delay mechanism is intended to provide the driver with a mechanism for batch servicing of interrupts.

2.5.3.2.2 Receive Status Queue almost full

The NICStAR can interrupt when the Receive Status Queue is 7/8 full thus providing a panic status notification. Normal processing of the queue via end of End-Of-PDU interrupts is expected to keep the queue below 7/8.

2.5.3.2.3 Receive Raw Cell

An Interrupt on reception of a Raw Cell can be generated when the general ***Raw Cell Queue Interrupt Enable*** bit in the ***Configuration*** register (Register #5) is asserted and the Raw Cell Interrupt Enable bit for the specific connection is asserted in the ***Receive Connection Table***. In general this interrupt would be used for signalling the arrival of an OAM cell. The enable bit in the ***Configuration*** register is a global enable for the driver to use. The enable bit in the connection table is for connection specific operation.

2.5.3.3 Receiving cells on a connection

2.5.3.3.1 General

When a cell is received, the NICStAR first determines the state of the particular VC along which the cell was received (i.e., open, closed, or no VC established). It does this by first indexing the LSB (Least Significant Bit) portion of both the VPI and VCI fields of the received cell's header into the ***Receive Connection Table***. If the fields match a table entry, the NICStAR determines if the VC is open or closed. If the VC is open, the NICStAR transfers the cell payload to the appropriate Free Buffer in host memory. If the VC is closed, or if the fields do not match a table entry (i.e., no VC established), the NICStAR either transfers the complete received cell to the ***Raw Cell Queue*** or discards the received cell, depending upon how the ***VPI/VCI Error Cell Accept*** bit field in the ***Configuration*** Register (Register #5) is set.

2.5.3.3.2 AAL5/AAL0

Payload data is placed into memory by the NICStAR by placing the first byte in the least significant byte address and placing subsequent bytes into every increasing byte address positions. The NICStAR will access memory as 32-bit words (D Words).

As cells are received and the payload is placed into a buffer in Host memory and the CRC-32 algorithm is applied. When a cell arrives which has the End-Of-PDU bit set (appropriate bit in PTI field), the accumulated CRC-32 is compared to CRC-32 contained in the newly received cell. The ***Control*** and ***Length*** field is placed into memory as part of the last payload data.

On End-Of-PDU the NICStAR writes an entry into the ***Receive Status Queue*** to notify the event. An interrupt is optional as configured in the ***Receive Connection Table***.

The NICStAR presents an AAL5 or AAL0 cell payload to the PCI bus interface with the format shown below. In the case of the last cell of an AAL5 CS-PDU, the Control/Length word and the CRC-32 word are inserted into the locations of the last two words of the cell payload (shown at the bottom of the following illustration). Note that for this case, the Control octets (CPCS-UU and CPI) and the Length Indicator octets, as well as the CRC-32 value, are presented to the PCI interface with the same byte-orientation as the cell payload words (i.e., they are also presented to the PCI interface in little-endian format). Note that the Control/Length word and CRC-32 word are not used for AAL0 cells.

An AAL5 cell is received at the UTOPIA interface as follows:

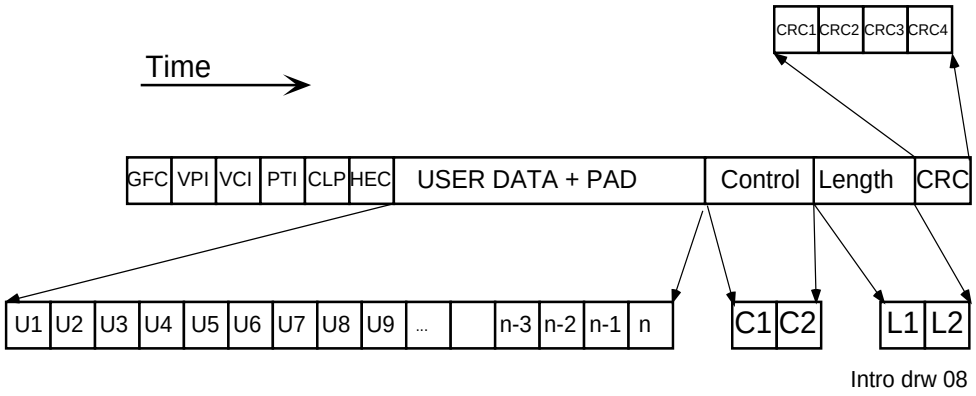


Figure 8: Received AAL5 Data Format at UTOPIA Interface- Last cell of the PDU

The NICStAR then presents an AAL5 cell to the PCI bus interface as follows:

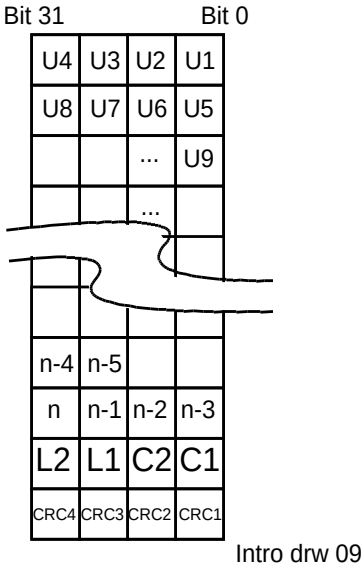


Figure 9: Received AAL5 Data Format at PCI Interface

2.5.3.3.3 AAL3/4

On reception of cells for AAL3/4 connections, the NICStAR separates out the Segment Type (ST), Sequence Number & Message Identification (MID), header fields, and the Length Indicator (LI) trailer field. These are placed in one 32-bit word after each 12 word payload. The device driver is responsible for checking the ST, SN and MID fields in the payload header, and the LI field in the payload trailer. The ST field specifies that the current cell is either the beginning of a message (BOM), a continuation of a message (COM), the end of a message (EOM), or that the current cell is a complete CS-PDU (i.e., Single Segment Message or SSM). The MID field is used to de-multiplex multiple, concurrent messages per VC connection; the MID field value is equal for all cells of a particular CS-PDU. The LI field specifies the length of the "real data" payload in bytes; this field is used to determine if the "real data" payload has been padded-out. It is possible that the second to the last cell in a PDU could also contain padding as well as the last cell in the PDU.

The bytes for the AAL3/4 cells are presented to the NICStAR as follows:

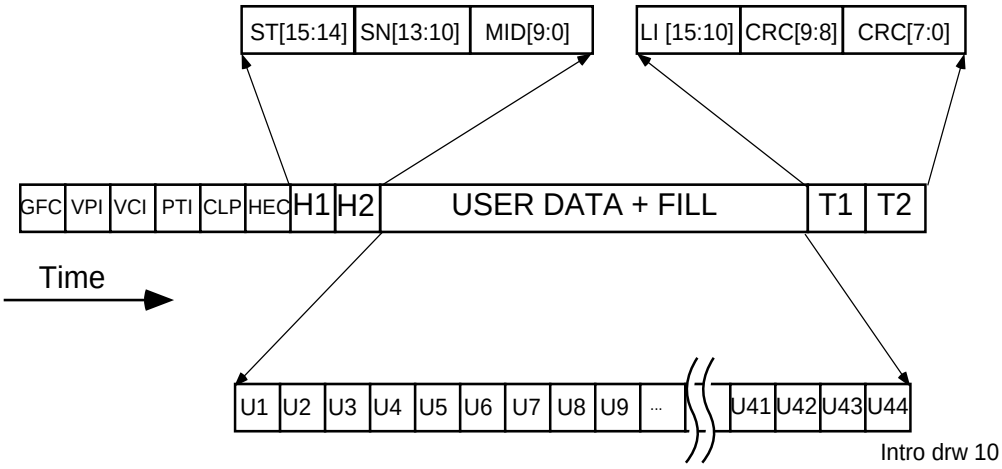


Figure 10: Received AAL3/4 Data Format at UTOPIA Interface

As the cells are received, they are placed into host memory using the PCI bus.

The following diagram shows how the words are presented to PCI bus: (Little Endian Mode see section 3.3 for further description regarding Big Endian Mode).

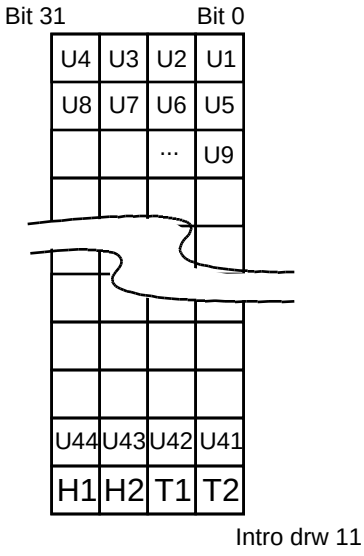


Figure 11: Received AAL3/4 Data Format at PCI Interface

2.5.3.4 Processing the Receive Status Queue

Buffers are passed off to the driver by placing an entry into the ***Receive Status Queue***. The driver processes the entries chaining buffers together as needed. As buffers are processed by the driver, Free Buffers should be added to the NICStAR in the ***Free Buffer Queues***.

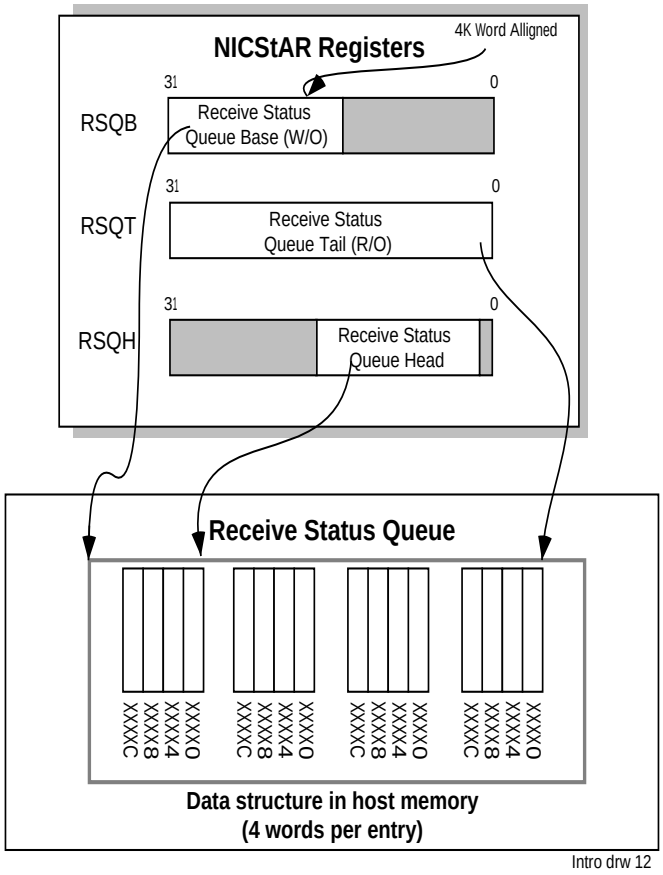


Figure 12: Receive Status Queue

It is intended that the driver first starts by initializing the ***Receive Status Queue*** to all zeros. When a ***Receive Status*** condition is sensed (by interrupt or timer based polling) the driver first starts by sampling the valid bit of the next possible status entry in the status queue. If it is valid, then the status entry is processed in order to "receive" the buffer, else return. If there is a valid entry in the queue and it has been processed and the valid bit cleared, then the driver should check for and process subsequent entries. When the driver determines there are no more entries in the ***Receive Status Queue***, it must write a head pointer to the ***Receive Status Queue Head*** register (Register #9). In this way the driver does not have to poll the NICStAR with expense (in terms of bandwidth) I/O operations to determine status buffer depth.

The intent of the ***Receive Status Queue Tail*** register (Register #8) is to facilitate

debug. Because of a FIFO buffer in the PCI interface for the receive path of the NICStAR, the queue tail register can be one entry ahead of what is in the CPU's local memory. This entry is "in-flight" across the PCI bus and will eventually find its way into memory.

2.5.3.5 Filling the Free Buffer Queue

As buffers are processed by the driver and free for use by the NICStAR, the driver must notify the NICStAR by placing a pointer on the ***Free Buffer Queue***. The driver can add up to two free buffer entries to the queue at a time via the command and data registers by monitoring the full and empty flags of the queues. The NICStAR can generate interrupt when either queue is empty. The driver loads the NICStAR's ***Data*** Registers and writes a Free Buffer command (***Write_FreeBufQ***) to the ***Command Register (***Register #4). When writing two entries the size of the buffers must be the same. The current length of the two ***Free Buffer Queues*** can be observed in the ***Status Register*** (Register #6).

2.5.3.6 Processing the Raw Cell Queue

When it is time to process the ***Raw Cell Queue***, the driver must calculate how many cells are in the queue to process. The driver can do this by comparing its Receive Raw Cell Head address to the ***Receive Raw Cell Queue Tail*** pointer (register #13) on the NICStAR. If the pointers are in the same buffer, then the driver processes the number of cells which is the difference of the tail and head. If the head pointer is not in the same buffer as the tail pointer, then the rest of the buffer contains valid cells. The host may then de-queue all of these cells. The "last" cell is a pointer (chain) to the next buffer to contain Raw Cells.

The ***Raw Cell Queue Tail*** pointer (Register #13) may precede the data in host memory by up to one (1) cell. This occurs because of a FIFO in the data path of the PCI interface. It is possible for the driver to read the tail pointer bypassing the cells ("in-flight") in the FIFO. When the NICStAR gets access to the PCI bus this cell will make it to the host memory. Therefore, the driver needs to be aware of this as it approaches the cell pointed to by the tail pointer. The driver could use a validation routine when it gets to this point. A validation could be done by waiting for up to 30us when it processes the last cell.

2.6 Segmenting ATM cells

Segmenting CS-PDU buffers and transmitting cells is controlled by the OS driver placing pointers to buffers ready for transmission into a segmentation channel queue. As buffers are transmitted status is return in a transmit status queue.

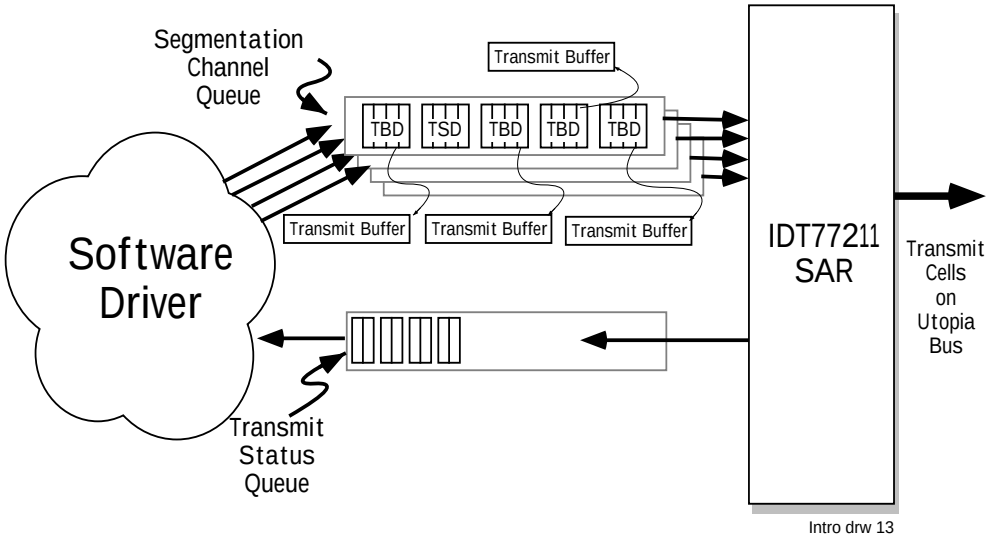


Figure 13: Segmentation Overview

The NICStAR provides for two basic mechanisms for controlling the transmission rate of cells. First, there is a Fixed Cell Rate group of channels which are controlled by a schedule table. Second, there is a Variable Cell Rate group of channels which are controlled by rate counters for connections that expect to vary their transmission rate "On-the-fly" as often as a buffer.

Buffers ready for transmission are identified by placing a ***Transmit Buffer Descriptor (TBD)*** on a ***Segmentation Channel Queue (TCQ)*** in CPU's local DRAM. The ***Segmentation Channel Queue (SCQ)*** is described by a ***Segmentation Channel Descriptor (SCD)*** kept in the NICStAR's local SRAM. Transmit status is placed in the ***Transmit Status Queue (TSQ)*** as a ***Transmit Status Request (TSR)*** is encountered in the ***Segmentation Channel Queue***.

A ***Transmit Schedule Table (TST)*** is used to schedule when each fixed rate channel will transmit a cell over the ATM network. It allocates the maximum bandwidth between channels. The schedule table is kept in the NICStAR's local SRAM.

2.6.1 Transmit Queues

2.6.1.1 Segmentation Channel

2.6.1.1.1 Segmentation Channel Descriptor

Segmentation Channel Descriptor (TCD) defines where in the CPU's local

memory a ***Segmentation Channel (SC)*** exists and what the state of the channel is. It is a twelve Dword structure located in SRAM. The first four words describe where the channel is located, the state of the tail and head pointers and the AAL5 CRC-32 if the channel is transmitting a AAL5 CS-PDU. The following eight words are divided into two four word structures which can cache up to two ***Transmit Buffer Descriptors (TBD)***.

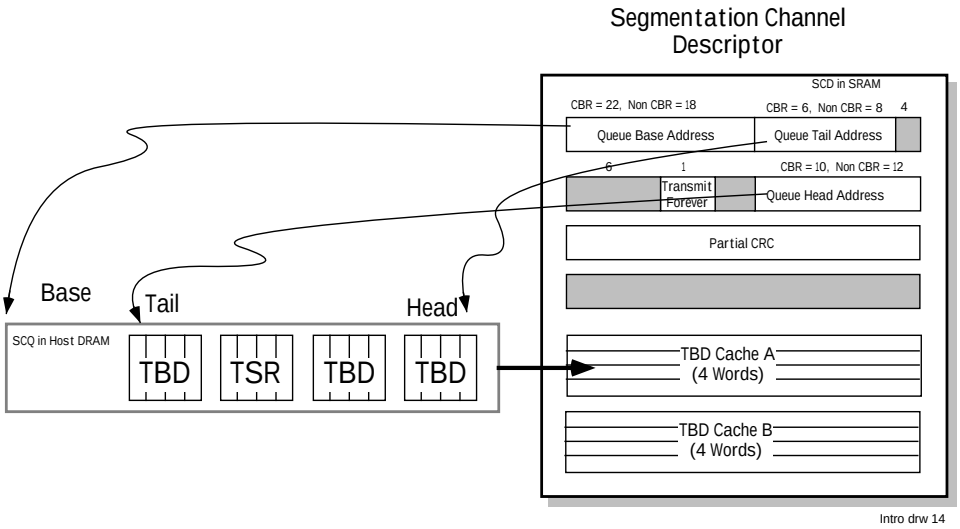


Figure 14: Segmentation Channel Descriptor

The format and function of the ***Segmentation Channel Descriptor*** depends on whether the channel being described is for a Fixed Cell Rate or Variable Cell Rate queue. The Base Address, tail and head pointers are different sizes accordingly.

Fixed Cell Rate Group

The ***Segmentation Channel Descriptor*** for the fixed rate group can be located anywhere in the SRAM that is allocated for purposes of the schedule table and channel descriptors. A Transmit ***Schedule Table*** entry which identifies a channel from this group, contains the address of the ***Segmentation Channel Descriptor***.

Variable Cell Rate Group

There are three Variable Cell Rate channels. Each of these channels has a specific hard-wired location in the SRAM. The ***Transmit Schedule Table*** entries which identify a cell from this group, identify this group as a whole. The rate counter associated with each of these channels is used to determine if a channel is ready to transmit a cell. A fixed priority encoder is employed to determine which channel will transmit in the cases where two or more channels are ready.

2.6.1.1.2 Segmentation Channel Queue

The ***Segmentation Channel Queue*** is located in the CPU's local DRAM. It is composed of multiple four-word entries which are either ***Transmit Status Request*** or ***Transmit Buffer Descriptors*** that identify buffers which are ready for segmentation. The queue size is 63 entries for each Fixed Cell Rate queue and 511 for each Variable Cell Rate queue.

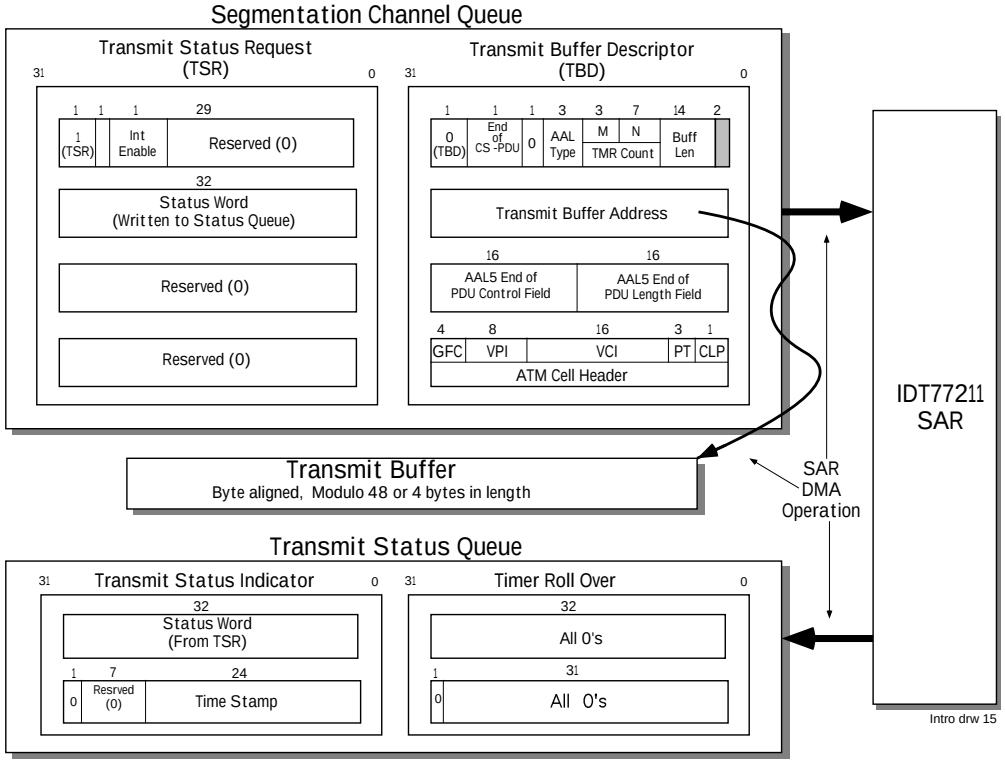


Figure 15: Segmentation Channel Queue Processing

The driver places ***Transmit Buffer Descriptor*** and ***Transmit Status Request*** in the queue as buffers are ready for transmission. As descriptors are added to the queue, the tail pointer in the ***Segmentation Channel Descriptor*** must be incremented by the driver to reflect such change. The NICStAR automatically increments the head pointer in the ***Segment Channel Descriptor*** as entries are consumed. The NICStAR will fetch these entries in a sequential order as they are placed in the queue. When putting multiple VC's in the same queue, all ***Transmit Buffer Descriptors*** belonging to the same CS-PDU must be grouped or continuous.

2.6.1.1.3 Transmit Buffer Descriptor

Transmit Buffer Descriptors are four 32-bit words which specify buffers to be segmented by he NICStAR. The first word is used to describe information such as: AAL type, End-Of-PDU, Interrupt preference, Rate timer (for Variable Cell Rate channels) and buffer length. ***Transmit Buffer Descriptors*** are identified by

a “0” in bit 31 of the first word.

The second word specifies a 32-bit physical address of where the buffer resides in the CPU’s local DRAM.

The third word specifies the first part of the AAL5 trailer which will be placed in the last AAL5 CS-PDU cell. The NICStAR does not modify this word and will place it in the transmit stream directly.

The final word is the 32-bit ATM cell header which is attached to each payload to form an ATM Cell as the corresponding buffer is segmented.

2.6.1.1.4 Transmit Status Request

Transmit Status Requests are four-word descriptors which direct the NICStAR to place a ***Transmit Status Indicator*** entry into the ***Transmit Status Queue***. One bit (31) of the first word is used to differentiate this entry between Transmit ***Status Request*** and ***Transmit Buffer Descriptor***. The second word is the ***Transmit Status*** word, which upon processing, is placed in the ***Transmit Status Indicator*** along with a time stamp. The NICStAR does not modify the ***Transmit Status*** word. Words 3 and 4 of the entry are not used by the NICStAR.

2.6.1.1.5 Variable Cell Rate m/n counter

Using the m/n rate counters, the driver can provide Variable Rate services like VBR, UBR and ABR. There are three m/n counters, each of which is associated with a variable rate segmentation channel. The m/n counter is used to control the rate at which the current buffer up for segmentation is going to be transmitted. The m/n counter value is associated with a buffer via the ***Transmit Buffer Descriptor*** data structure which contains a field for the m/n value.

The Counter n value specifies a divisor of the basic cell line rate. It is a 7-bit field, and thus provides 127 integer options from 1 to 127. The Counter m value is specified in a 3-bit field, and thus provides 7 integer options from 1 to 7 which specifies how many cells will be burst back to back when the n counter rolls over. M/N must be equal or less than 1 (m/n <=1). These fields specify the maximum data rate for a VC, which may be any speed from 1.22 Mbps (m/n = 1/127) to 155.52 Mbps (m/n = 1/1) for 155.52 Mbps ATM. The following rate calculation is based on a SONET framer PHY device.

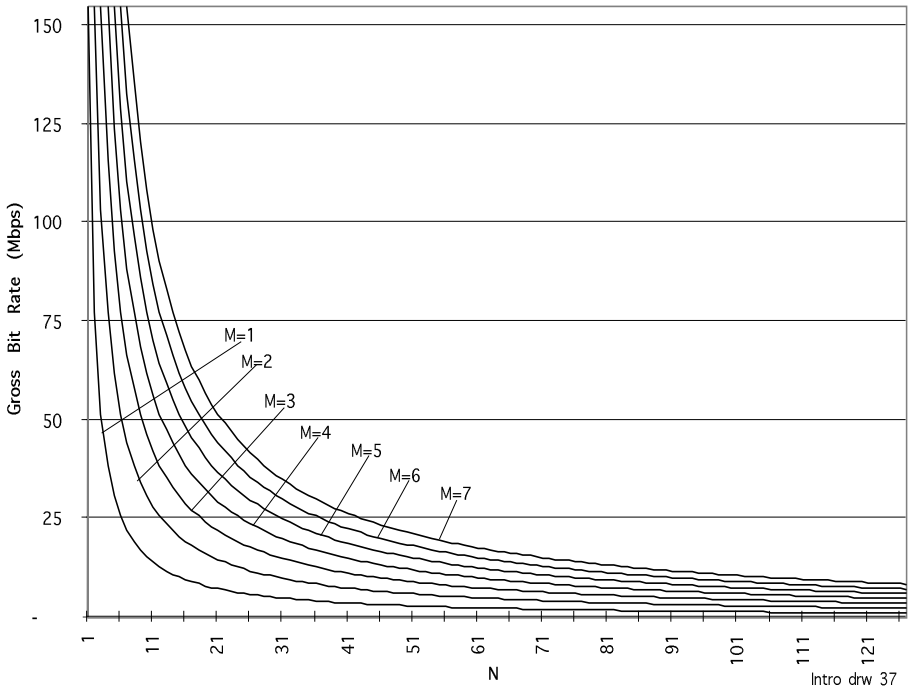
Gross bit rate= LR * (m/n),

Net cell rate= LR (1 - SONET OH) / (424bits/cell)* (m/n),

where LR = line rate, m is the numerator and n is the denominator.

SONET OH = SONET framing overhead (TOH and POH)

The SAR will transmit, when requested, m cells and then count down n cell times before transmitting cells from this VCI/VPI again. As the SAR transmits the last cell in the PDU, the SAR will get the next PDU to transmit. Getting this new PDU will impose a new m/n value and begin transmitting prior to the previous n counter reaching zero.



The M/N values plotted as a function of gross bit rate (Mbps).

The table below illustrates the general principle of using the rate counters (please note that this table includes only a small subset of all available data rate options).

Table 1: Variable Cell Rate Setting

Gross Bit Rate (Mbps)	Net Cell Rate (1,000 cell/s)	m Counter Value	n Counter Value
155.52	353.21	1	1
136.08	309.05	7	8
133.30	302.75	6	7
129.6	294.34	5	6
124.42	282.57	4	5
116.64	264.91	3	4
103.68	235.47	2	3
77.76	176.61	1	2
1.22	2.78	1	127

The value of "m" or "n" should never be set to zero or un-predictable operation will result.

The position of cells for Variable Rate services will be effected by the Transmit Schedule Table with regards to what time slots have been allocated for Variable Rate services. If there is not enough bandwidth specified by the table, then the table will set the upper limit on the Variable Rate services rate.

2.6.1.2 Transmit Status

The NICStAR writes a ***Transmit Status Indicator (TSI)*** which is a two word entry in the ***Transmit Status Queue (TSQ)*** located in the host memory as a result of one of the two conditions: a ***Transmit Status Request (TSR)*** was encountered in a ***Segmentation Channel Queue***, or a ***Time Stamp Counter*** rolls over. The content of the first word is dependent on the condition that caused the ***Transmit Status Indicator*** to be written to the host memory. If this ***TSI*** is the result of a ***TSR***, then the first word will contain the 32-bit status word pulled from the ***TSR***. If the ***TSI*** is a result of a timer roll-over condition, then the first word will be cleared to all zeros. The second word of the ***TSI*** will contain the value of the Time Stamp Counter at the point in time when the NICStAR processed the ***TSR***.

The ***Transmit Status Queue*** is written by the NICStAR and read (consumed) by the OS device driver. Handling the ***Transmit Status Queue*** can be made to happen very efficiently by reading only the CPU's local memory. One bit in the second word of the ***Transmit Status Indicator*** entry is the **empty** bit for that entry. This **empty** bit will be set to "1" by the driver during initialization to indicate that this entry is not valid. After the NICStAR fills an entry ***Transmit Status Indicator*** with valid information, the **empty** bit will be set to "0" by the NICStAR to indicate a valid entry. The driver can monitor NICStAR progress by polling the **empty** bit (in host memory) without reading from the NICStAR across the PCI bus. Traditional methods of polling the tail and head pointers of the NICStAR can use up many valuable cycles on the PCI bus.

The size of the ***Transmit Status Queue*** is fixed at 8K bytes (1024 entries). The location is determined by the ***Transmit Status Queue*** Base register. This address must be a multiple of 8K bytes.

2.6.1.2.1 Transmit Time Stamp Counter

The transmit ***Time Stamp Counter*** is a 24-bit counter that is incremented by 666 divided by the NICStAR clock. For a 50MHz SAR clock, an interrupt would occur approximately every 3.3 minutes. It is used to provide a time stamp for all entries placed in the ***Transmit Status Queue***. It can be read by the NICStAR and thus provide a real time count for measuring other operations of the NICStAR. It is programmable through the ***Configuration Register*** to have the NICStAR generate an interrupt when the ***Time Stamp Counter*** rolls over. If transmit is disabled, the ***Time Stamp Counter*** is frozen and no roll over will result.

2.6.2 Transmit Schedule Table

The ***Transmit Schedule Table*** is used to schedule each channel when to transmit a cell over the ATM network. It also allocates the maximum bandwidth among channels. The schedule table is kept in the NICStAR's local SRAM.

Access to the ***Transmit Schedule Table*** is intended to be infrequent, and is accomplished via the SRAM read/write commands. As Fixed Cell Rate channels are opened, the ***Transmit Schedule Table*** will be modified to reflect such change.

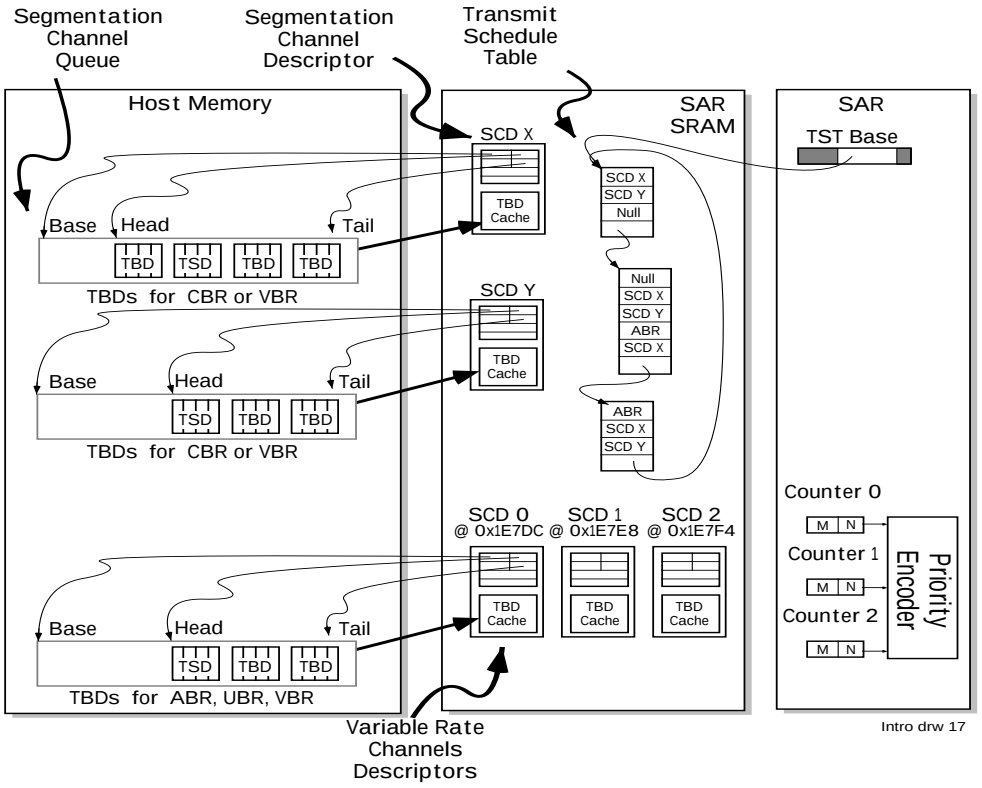


Figure 16: Transmit Schedule Table

The table consists of a sequence of 32-bit single word entries and is "executed" in sequence like a program instructing the NICStAR what and when to do with the entry regarding transmitting cells to the ATM network. Each entry specifies one of four actions:

i) Transmit a cell from a specific ***Segmentation Channel***,

ii) Transmit a cell from the Variable Cell Rate channel group,

iii) Transmit a NULL cell, or

iv) Jump to a new location.

When a segmentation channel is specified explicitly by the ***Transmit Schedule Table*** and the channel can not transmit a cell (i.e. the channel is empty), then the Variable Cell Rate channel group is scanned for a possible cell to transmit. When a transmit from the Variable Cell Rate channel group is specified and the channel can not transmit a cell, then a NULL cell is transmitted.

2.6.3 Transmit Operations and Algorithms

The NICStAR architecture is optimized with the intent such that the NICStAR moves large blocks of data in PCI Bus Master mode and the CPU performs less frequent accesses that must be required by the CPU. Access from the host CPU to local host memory could be as much as 5 to 10 times faster than accessing the NICStAR on the PCI bus. In most cases, handling of queues can be performed in several ways. The algorithm outlined in this section are suggestions which are intended for optimizing data movement and system performance.

2.6.3.1 General transmit actions

When the transmit section of the NICStAR is enabled for the first time, the NICStAR reads the ***Transmit Schedule Base*** register and starts "execution" from that point. Each entry in the ***Transmit Schedule Tables*** specifies one of four actions: transmit a cell from a specific segmentation channel, transmit a cell from the Variable Cell Rate channel group, transmit a NULL cell, or jump to a new location. When a segmentation channel is specified explicitly as Fixed Rate Channel in ***Transmit Schedule Table*** and the channel can not transmit a cell (i.e. the channel is empty), then the Variable Cell Rate channel group is scanned for a possible cell to transmit. When a transmit from the Variable Cell Rate channel group is specified in ***Transmit Schedule Table*** and none of the channels can transmit a cell, then a NULL cell is transmitted.

The NICStAR reads ahead in the table and issues fetches for cell payloads on the PCI bus as 12 word PCI bus master transfers. As cell payloads are fetched and brought into the NICStAR, complete ATM cells are constructed by attaching the 32-bit word header contained in the ***Transmit Buffer Descriptor*** and a place holder for the HEC byte. The value of the HEC byte is calculated and replaced outside the NICStAR by the PHY device.

2.6.3.1.1 AAL5

Payload data is read from the host memory in burst of twelve 32-bit words. As the words are fetched, the byte with the lowest PCI byte address is sent out the Utopia interface first (Little endian).

As payloads are fetched from Host memory the CRC-32 algorithm is applied. When a last payload arrives which has the End-Of-PDU bit set, The ***AAL5 Control*** and ***Length*** fields of the last ***Transmit Buffer Descriptor***, as well as the accumulated CRC-32 are all placed as the AAL5 trailer in the last cell.

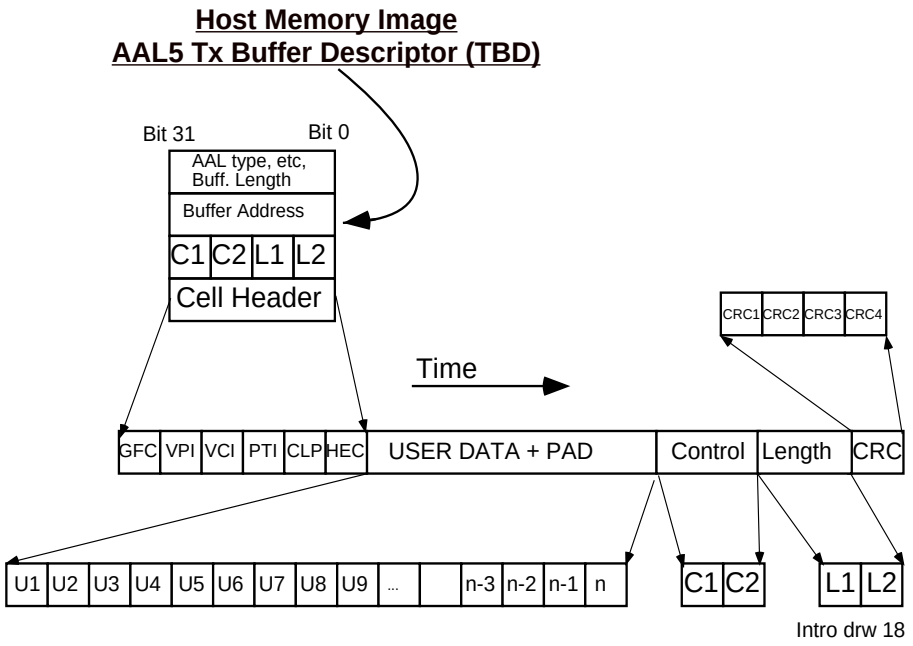


Figure 17: Transmit AAL5 Data Format (For last cell in PDU) at UTOPIA Interface

The NICStAR then fetches an AAL5 cell from the PCI bus interface as follows: (Little Endian Mode see section 3.3 for further description regarding Big Endian Mode)

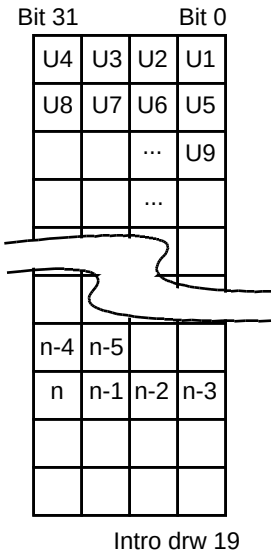


Figure 18: Transmit AAL5 Data Format at PCI Interface

The PCI bus is considered a little endian bus. It is expected that the Host Processor will have its own translation for the Host Memory to the PCI bus.

2.6.3.1.2 AAL3/4

AAL3/4 data to be segmented and transmitted by the NICStAR must be formatted as shown below before it is queued for transmission. In addition, the device driver must "fill in" the Segment Type (ST), Sequence Number (SN) and Message Identification (MID) bit fields of the payload header, and the Length Indicator (LI) bit field of the payload trailer. The NICStAR will calculate the CRC-10 value as it processes the cell, and insert its calculated CRC-10 value into the CRC-10 bit field of the payload trailer

The NICStAR expects an AAL3/4 cell payload to be represented with the following format at the PCI bus interface as memory is read. (This is for Little Endian Mode see section 3.3 for further description regarding Big Endian Mode)

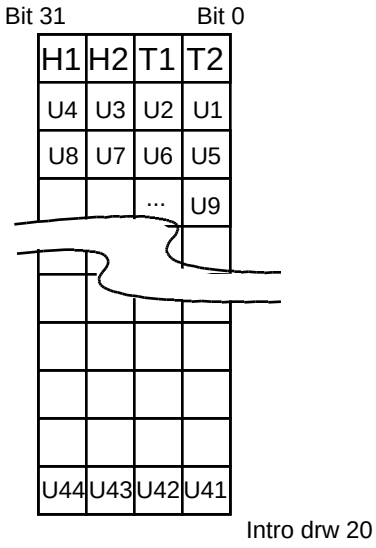


Figure 19: Transmit AAL3/4 Data Format at PCI Interface

The NICStAR then presents the cells to the Utopia interface in the following order:

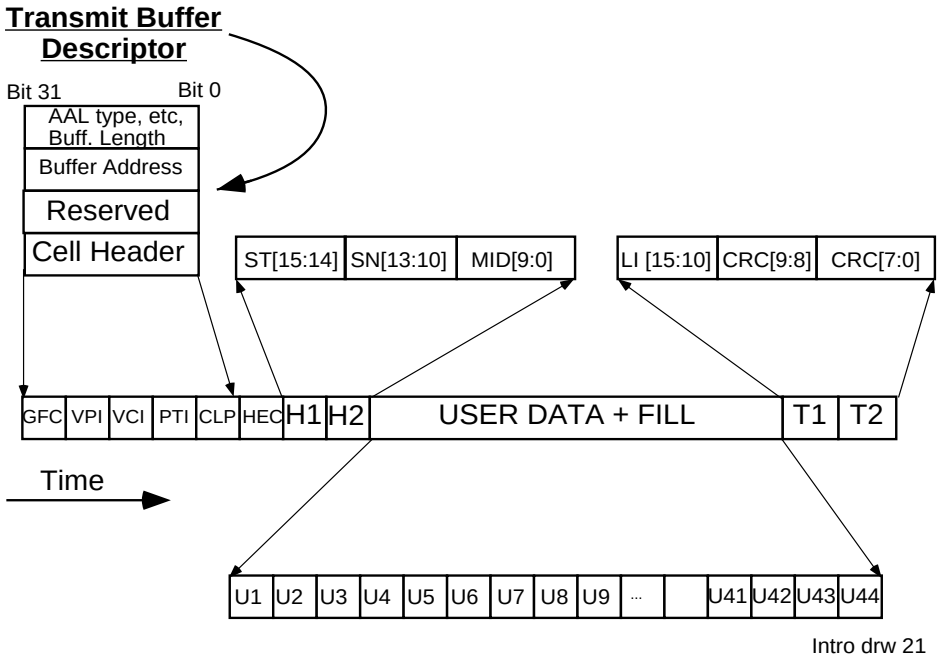


Figure 20: Transmit AAL3/4 Data Format at UTOPIA Interface

2.6.3.2 Guaranteeing Cell Delay Variation (CDV)

Minimal Cell Delay Variation (CDV) is very important for CBR ATM services. CDV is the variation in cell arrival time from the expected arrival time. Many scheduling mechanisms use rate timers and deal in units of cells per second. In the NICStAR, time is dealt with in terms of cell slots on the physical media as defined by the PHY. This means that there is never a round-off error of when to ship a cell which would result in jitter and thus increased CDV. The added advantage is that the NICStAR can be used with many different physical line rates without special tuning. Another advantage is that the ATM interface requires only one very accurate oscillator (the PHY's) as opposed to an additional one for the NICStAR's rate timers. The SAR is self timed or regulated by the Phy's back pressure. UBR and VBR services do not have the same stringent requirement of low CDV, therefore a trade-off of CDV performance for a CBR connection can be made against that of a UBR or VBR connection.

Using the PHY's very accurate transmit bit rate clock with back pressure on a cell by cell basis to the NICStAR and the ***Transmit Schedule Table***, a very low CDV for specific Fixed Rate segmentation channels at the point of transmission to the ATM network can be assured. As the PHY needs a new cell from the NICStAR for transmission, it signals the request to the NICStAR with a Cell Available (CLAV) signal. The NICStAR always fetches cell payloads ahead of time and places them in a transmit ready queue. As empty cell locations open in the transmit ready queue, the ***Transmit Schedule Table*** state machine in the NICStAR advances and fetches a new cell payload to be placed in the queue. In this way the PHYs very accurate transmit clock throttles rate of cell transmission to a very accurate rate for the Fixed Cell Rate channels (used for CBR).

2.6.3.3 Handling PCI Latency

One of the factors in a computer system that can profoundly effect CDV is the latency on the computer system's main bus which transports cell payloads between memory and the NICStAR. Whenever the NICStAR requests access to the bus, there is variable (and unpredictable) delay before the transfer of the payload. To handle this condition, the NICStAR fetches cells ahead of time and builds a reserve of cells in a transmit ready queue in anticipation of possible latency in acquiring the PCI bus. If the transmit ready queue in the NICStAR is emptied due to excessive PCI latency, then NULL cells are generated across the interface to the PHY in place of cells with useful payloads. NULL cells transmitted due to queue empty condition are counted. If this happens, the NICStAR will skip all entries which specify Variable Rate channel group and NULL cells in the ***Transmit Schedule Table*** and only "execute" entries which are specified as Fixed Rate channels. As entries are skipped the count of transmitted NULL cells due to the empty queue condition is decremented. When the count reaches zero, further skipping of entries is terminated. In this way the NICStAR "catches up" on the Fixed Rate channels. Many ATM networks may not drop the cells during the "catch up" period because of finite CDV Tolerance (CDVT). CDVT is a negotiated connection parameter.

When the transmit ready queue inside the NICStAR runs empty due to latency on the PCI bus, CDV performance may be violated. Guaranteeing of the PCI bus bandwidth to meet the Fixed Rate channel requirement is the responsibility of the user and the OS drivers. This mechanism allows for a more "graceful" degradation without a sharp break in service.

2.6.3.4 Transmit Interrupts

The NICStAR has multiple interrupt sources as a result of transmit operations. When an interrupt is asserted internally in the NICStAR, an interrupt is posted to the Host CPU via the PCI bus. The driver must then read the ***Status*** register (Register #6) to determine the cause of the interrupt. The driver may then disable a specific interrupt source by masking off the appropriate bit in the ***Configuration*** register (Register #5). After the appropriate interrupt source has been serviced, the driver can un-mask that interrupt again.

2.6.3.4.1 Transmit Status Queue entry

The NICStAR is capable of asserting an interrupt each time a ***Transmit Status Indicator*** entry is posted in the ***Transmit Status Queue***. Entries in the status queue are the drivers main way of knowing what progress the transmitter has made. The driver must put a ***Transmit Status Request*** in the ***Segmentation Channel Queue*** in order to have an indicator placed in the status queue.

It is recommended that if TSR interrupts are requested, they be requested for all TSRs. Likewise, if TSRs do not request interrupts, it is recommended that they are not requested for all TSRs. If there is a change from TSRs requesting interrupts to TSRs not requesting interrupts - the first TSR that does not request an interrupt will function as though an interrupt has been requested. The first non-interrupted TSRs will function without an interrupt if proceeded by a non-interrupt TSR.

2.6.3.4.2 Transmit Interrupt (TxINT)

If this function is enabled in the ***Configuration*** register (Register #5), the NICStAR will assert an interrupt to the Host CPU to indicate that a TSI has been written in Host memory.

2.6.3.4.3 Time Stamp Counter Roll over

The NICStAR has a ***Time Stamp Counter*** which is used to make entries in the ***Transmit Status Queue***. When the ***Time Stamp Counter*** reaches maximum value, it will roll over and posts an entry in the ***Transmit Status Queue***. When an entry is posted it an interrupt is asserted to mark the event. The interrupt is generated to the Host CPU if the Roll Over Interrupt enable bit is set in the ***Configuration*** register (Register #5). If transmit is disabled, the ***Time Stamp Counter*** is frozen and no roll over will result.

2.6.3.4.4 Transmit Status Queue almost full

When the ***Transmit Status Queue*** is 7/8 full, an interrupt is asserted to the host CPU if its interrupt enable bit is set in the ***Configuration*** register (Register #5).This is meant to be a panic condition notification to the driver that the status queue has reached a critical stage and needs a higher priority servicing.

2.6.3.5 Processing the Transmit Status Queue

The ***Transmit Status Queue*** is written by the NICStAR and read (consumed) by the OS device driver. Handling the ***Transmit Status Queue*** can be very efficient by reading only the CPU's local memory. One bit in the second word of the ***Transmit Status Indicator*** entry is the **empty** bit for that entry. This **empty** bit will be set to "1" by the driver during initialization to indicate that this entry is not valid. After the NICStAR fills an entry ***Transmit Status Indicator*** with valid information, the **empty** bit will be set to "0" by the NICStAR to indicate a valid entry. The driver can monitor NICStAR progress by polling the **empty** bit (in host memory) without reading from the NICStAR across the PCI bus. Traditional methods of polling the tail and head pointers of the NICStAR can use up many valuable cycles on the PCI bus.

When the NICStAR processes a timer roll over, it writes a word of all zero's into the first word of the queue entry. Thus the driver can distinguish valid entry between a roll over stamp and an entry made as a response to a ***Transmit Status Request*** in the ***Segmentation Channel Queue***.

When it is time to process the ***Transmit Status Queue*** which could be determined by interrupt or time out, the software should test the ***empty*** bit at the location identified by the software's head pointer. If the ***empty*** bit is cleared, then the software should process entry. Upon processing the entry, the software must increment the head pointer and test the ***empty*** bit of the next entry. At this point if the ***empty*** bit is set, then the driver must write the software's head pointer to the NICStAR's ***Transmit Status Queue Head (Register #18)*** pointer to tell the NICStAR that entries in the queue have been freed. This method avoids costly PCI bus reads of the NICStAR's status and does only one PCI bus write, which is typically very fast because of posting the write to a write buffer.

2.6.3.6 Changing Transmit Schedule Tables

When transmission is enabled for the first time the NICStAR reads the ***Transmit Schedule Base*** (Register #15) register and starts "execution" from that point. The ***Transmit Schedule Base*** is read only once by the NICStAR and is never used again until reset occurs.

The ***Transmit Schedule Table*** is a sequence of "instructions" to the NICStAR on what to transmit. The table can be made with multiple jumps. The last entry in the table must be a jump to the start of the table. To change to a new schedule table, a jump link must be modified to "point" to a new table. The NICStAR will then follow the link into the new table. The new table should then have a link back to the top of the new table. Once the jump is taken the old table can be modified.

2.7 PCI Registers and Interface

The NICStAR contains two sets of PCI registers for the host interface, they are the PCI Configuration Registers and the network operational registers. In addition, the host accesses all other resources associated with the NICStAR through the PCI interface.

2.7.1 PCI Configuration Registers

There are 256 bytes in the configuration block of PCI Configuration Space Registers. The host system BIOS software reads the Configuration Space Registers using the Configuration Read command when the IDSEL# and AD[7:2] signals. (AD[1:0] = 00 and AD[31:8] are ignored). The BIOS then determines the NICStAR I/O Base Address and the NICStAR Memory Base Address, which the host uses to access to the NICStAR's "operational" registers.

2.7.2 NICStAR Network Operation Registers (Registers #0 thru 20)

The NICStAR network operation registers are decoded as a 4K bytes of memory address space. They may be mapped as both a memory addressed device and as an I/O addressed device through the PCI bus interface. The NICStAR supports 32-bit PCI and does not support a target or initiator 64-bit accesses. Host writes to the reserved address space has no effect, however, the NICStAR signals handshake with the PCI bus as if a write has taken place. Host reads from the reserved addresses will be returned with zero data.

2.7.3 PCI Expansion EPROM

An expansion EPROM is memory mapped in the PCI memory address space. The NICStAR reads the 8-bit physical EPROM from the SRAM memory address space. The NICStAR accesses the EPROM four times making up one 32-bit word. The NICStAR presents the PCI bus with 32 bit read data. When the NICStAR is accessing the EPROM 8 bit at a time, the PCI bus TRDY# is de-asserted. With the PCI bus running at maximum speed of 33Mhz, an EPROM of 70ns is required.

To read the EPROM, the transmitter and receiver functions must be disabled. This is necessary since the EPROM is mapped into main memory and timing is controlled by the PCI master. A read from this memory space results in a PCI read of the SAR by the PCI controller. The transaction is initiated by the controller asserting FRAME for one cycle and monitoring the TRDY signal from the SAR indicating the action is complete. The SAR will break down the double word read command into four successive byte reads and generate an address to the EPROM for each byte. The address will be valid for at least

three PCI clock cycles to ensure meeting maximum propagation delays of the eprom. The first and last byte reads consume four clock cycles of the PCI bus to accommodate the propagation delays through the SAR.

EPROM critical timing is determined by the second and third byte transfers only. During these transfers the address to EPROM is generated off the rising edge of PCI clock and the data is latched on the rising edge of PCI clock three cycles later. The maximum propagation delay for the EPROM is three cycles of PCI clock minus Address generation in the SAR (20ns max) TAA= 3*Tcyc - 20ns = 70ns.

2.7.4 PCI Bus Interface

As a PCI bus master, the NICStAR supports only memory accesses. As a PCI slave, it supports memory, I/O and configuration accesses. The NICStAR performs all key data transfers to and from host memory (or PCI peripherals) as burst transfers with the NICStAR as a PCI bus master.

The NICStAR supports 32-bit PCI and will not initiate a 64-bit transfer. NICStAR will inter-operate with 64-bit PCI devices by doing 32-bit transfers. Host writes to the reserved address space has no effect but the NICStAR signals handshaking with the PCI bus as if a write is taking place. Host reads from the reserved addresses will be returned with zero data.

2.7.4.1 Memory Accesses as a PCI Master

The NICStAR supports the PCI Memory Write, and PCI Memory Read, and PCI memory read line commands. For the PCI Memory Write command, it writes to an agent mapped in the memory address space, while for the PCI Read command, it reads from an agent mapped in the memory address space.

2.7.4.2 I/O Accesses as a PCI Master

The NICStAR does not support the PCI I/O Write nor PCI I/O Read commands

2.7.4.3 Memory Accesses as a PCI Slave

The NICStAR supports both the PCI Memory Write and PCI Memory Read commands. For the PCI Memory Write command, the NICStAR is written to as an agent mapped in the memory address space, while for the PCI Memory Read command, the NICStAR is read from as an agent mapped in the memory address space.

2.7.4.4 I/O Accesses as a PCI Slave

The NICStAR supports both the PCI I/O Write and all PCI I/O Read commands. For the PCI I/O Write command, the NICStAR is written to as an agent mapped in the I/O address space; while for the PCI I/O Read command, the NICStAR is read from as an agent mapped in the I/O address space.

3 NICStAR Data Structures and Registers

This section describes the data structures associated with the NICStAR and its internal registers. They are covered in the following sub-sections:

NICStAR Data Structures
Local SRAM Memory Maps
AAL Data Formats in Host memory
NICStAR Network Operation Registers
NICStAR PCI Configuration Registers

3.1 NICStAR Data Structures

The NICStAR requires data structures set up external to the device to support its transmit and receive operations. Some data structures are located in the external local SRAM while others are located in the host memory.

Some of these data structures are shared between the device driver and the NICStAR. Device driver can access the local SRAM indirectly through the NICStAR registers (described in NICStAR Register section). Likewise, the NICStAR can access the host memory by using its PCI bus master capability.

3.1.1 Data Structure for Receive Section

The data structures for the Receive Section include the following:

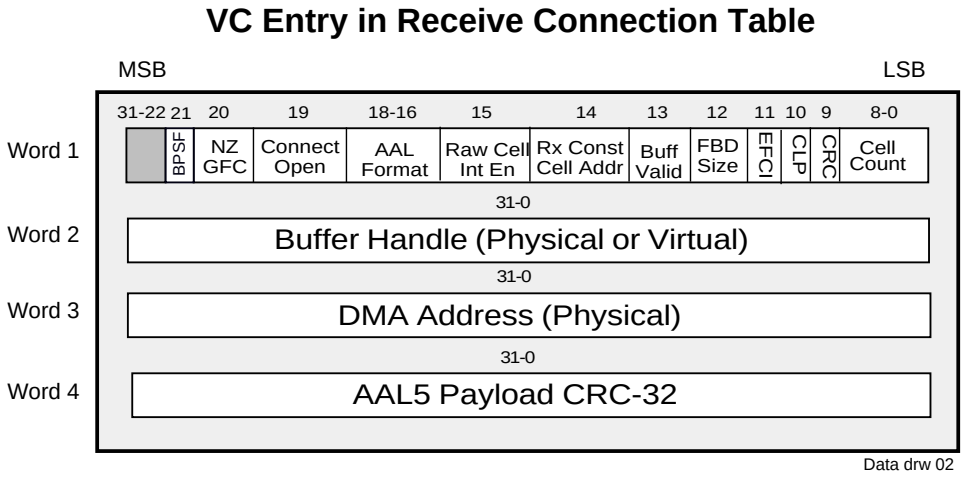
Table 2: Location of Receive Data Structures

Name of data structure	Location	Size (Bytes)
Receive Connection Table (RCT)	local SRAM	64K – 256K
RCT Entry (RCTE)	local SRAM	16
Free Buffer Descriptor (FBD)	local SRAM	8
Receive Status Queue (RSQ)	host memory	2K – 8K
RSQ Entry (RSQE)	host memory	16
Raw Cell Queue (RCQ)	host memory	2K to 8,192K (Large Free Buffers)

3.1.1.1 Receive Connection Table RCT)

The Receive Connection Table (RCT) is located in local SRAM and contains four words of information for each established VC.

3.1.1.2 Receive Connection Table Entry(RCTE)



3.1.1.2.1 Receive Connection Table Entry (RCTE): Word 1 of 4

Receive Connection Table (RCT) Entry

RCTE	BIT FIELDS												
	31-22	21	20	19	18-16	15	14	13	12	11	10	9	8-0
Word 1 of 4	RSVD	BPSF	NZ GFC	Connect Open	AAL Format	Raw Cell Int En	Rx Const Cell Addr	Buff Valid	FBD Size	EFCI	CLP	CRC	Cell Count

BIT FIELD	FUNCTION	DESCRIPTION
31-22	RSVD	Reserved. Set to zero.
21	BPSF	Bypass Small Free Buffers, if set to 1, use only Large Free Buffers.
20	NZ GFC	Non zero GFC. If set to 0, the GFC field was zero for all received ATM cells along this VC. If set to 1, one or more received ATM cells along this VC had a non-zero GFC field.
19	Connect Open	If set to 1, it indicates that the VC is open and that received ATM cells from this channel will be reassembled into host memory. The device driver is responsible for setting this bit in all the table entries using the Open Command and for maintaining the state of all current open connections. This field must be initialized by the host driver. If set to 0, it indicates connection not open.

Data drw 03

Receive Connection Table (RCT) Entry, Word 1 of 4 - continued

BIT FIELD	FUNCTION	DESCRIPTION
18-16	AAL Format	AAL Format. Set by the device driver to identify which AAL format (or none) to use for reassembly. 000 = AAL0; 001 = AAL 3/4; 010 = AAL5; 011 = Raw Cell Queue; 100 = Reserved; 101 = Reserved; 110 = Reserved; 111 = Reserved. This field must be initialized by the host driver.
15	Raw Cell Int En	Raw Cell Interrupt Enable. Set to "1" by the device driver, to cause the NICStAR to generate an interrupt after a "Raw Cell" is stored into host memory. If set to "0", no interrupt will be generated after a "Raw Cell" is stored into host memory. The Raw Cell Interrupt Enable is typically used for OAM F4 or Resource Management cells. Note that this bit field specifies interrupt enable per VC; the Raw Cell Interrupt Enable bit field of the Configuration Register (bit 15), which specifies global interrupt enable, must be set for any per VC interrupt enable to have an effect.

Data drw 04

Receive Connection Table (RCT) Entry, Word 1 of 4 - continued

BIT FIELD	FUNCTION	DESCRIPTION
14	Rx Const Cell Addr	Rx Constant Cell Address. If equal to 0 , received ATM cells are processed normally (i.e., cell payloads are reassembled directly into host memory buffers; when a buffer may not accommodate another complete 12 word cell payload, the NICStAR "pulls" a FBD from the Large Free Buffer Queue and begins writing new cell payloads into the new buffer). If equal to 1, the NICStAR stores all cell payloads to the same 12 word memory region (i.e., the NICStAR always presents the same DMA addresses to the PCI bus; the NICStAR does not increment the DMA address it presents to the PCI bus for the second or subsequent received cell payloads of a specific VC). This bit field is written by the host and read by the NICStAR.
13	Buff Valid	Buffer Valid. When this bit is set to 0, the Buffer Handle is not valid. When this bit is set to 1, the Buffer Handle is valid, and ATM cell payloads may be stored into the corresponding host memory buffer. This bit is initially set to zero by the device driver, and then set to one by the NICStAR after a free buffer has been "pulled" from one of the free buffer queues (just after the NICStAR receives the first cell of a CS-PDU for the particular VC). This bit is cleared to zero whenever a buffer is filled (i.e., it may not accommodate another complete cell payload) or an End_of_CS-PDU condition is detected.
12	FBD Size	FBD Size. 0 = The current buffer was obtained from the Rx **Small** Free Buffer Queue. 1 = The current buffer was obtained from the Rx **Large** Free Buffer Queue. This bit field is written by the NICStAR and read by the host.

Data drw 05

Receive Connection Table (RCT) Entry, Word 1 of 4 - continued

BIT FIELD	FUNCTION	DESCRIPTION
11	EFCI	EFCI Congestion Flag. 0 = No congestion detected. 1 = Congestion detected. If the PTI field of the received ATM cell header is either PTI = 010 or = 011, then the NICStAR sets the congestion flag equal to one. After the complete CS-PDU has been reassembled, the NICStAR clears the flag to zero. Under normal operation, this bit equals to zero, which indicates no congestion detected.
10	CLP	Cell Loss Priority Flag. 0 = Received cell with CLP bit = 1 was **NOT** detected. 1 = Received cell with CLP bit = 1 was detected. If the CLP (Cell Loss Priority) bit of the received ATM cell header is set to one, then the CLP flag is set to one. After the complete CS-PDU has been reassembled, the NICStAR clears the flag to zero.
9	CRC	Received CRC Error. For AAL5 (CRC-32) and AAL3/4 (CRC-10) only. 0 = No CRC error detected at end of received CS-PDU. 1 = CRC error detected at end of received CS-PDU. This bit field is written by the NICStAR.
8-0	Cell Count	Cell Count. Specifies the number of ATM cell payloads which have been reassembled into the current buffer, where each cell payload represents 12 words. The maximum size of a Large Free Buffer is 16K bytes, thus the maximum number of cells per buffer is 341. This bit field is written by the NICStAR.

Data drw 06

3.1.1.2.2 Receive Connection Table Entry (RCTE): Word 2 of 4

Receive Connection Table (RCT) Entry

RCTE	BIT FIELDS
Word 2 of 4	31-0
	Buffer Handle

BIT FIELD	FUNCTION	DESCRIPTION
31-0	Buffer Handle	Buffer Handle. This bit field is written by the NICStAR. The "virtual" start address of a free buffer in host memory. This value is written by the device driver into a free buffer queue in local SRAM, and fetched by the NICStAR into the Receive Connection Table when needed by a VC.

Data drw 07

3.1.1.2.3 Receive Connection Table Entry (RCTE): Word 3 of 4

RCTE	BIT FIELDS
	31-0
Word 3 of 4	DMA Address

BIT FIELD	FUNCTION	DESCRIPTION
31-0	DMA Address	DMA Address. This bit field is written by the NICStAR. The "physical" address of the starting location of the free buffer in a host memory. This value is updated by the NICStAR as ATM cell payloads are stored into host memory except when bit 14 of word 1 is set.

Data drw 08

3.1.1.2.4 Receive Connection Table Entry (RCTE): Word 4 of 4

Receive Connection Table (RCT) Entry

RCTE	BIT FIELDS
	31-0
Word 4 of 4	AAL5 CRC-32

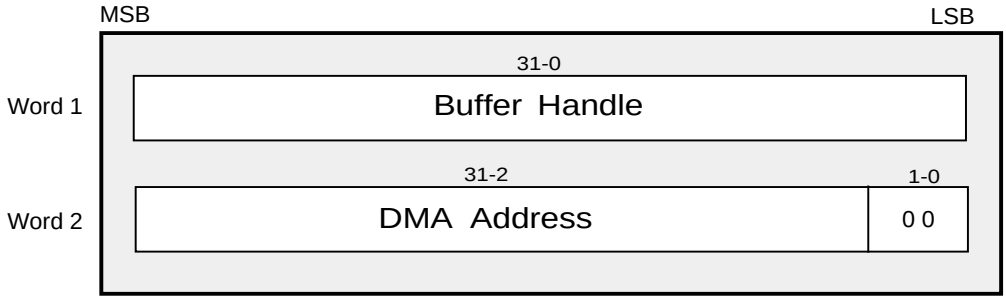
BIT FIELD	FUNCTION	DESCRIPTION
31-0	AAL5 CRC-32	AAL5 Payload CRC-32. This bit field is used for AAL5 type only. It is the current "running total" of the CRC-32 calculation for the AAL5 cell payloads of a CS-PDU. The value of this bit field is updated by the NICStAR as each new ATM cell is received. The final CRC-32 value will be written by the NICStAR into the word 3 location in the Receive Status Queue entry for this CS-PDU. This word must be set to "FFFF FFFF" during initialization by the host. The NICStAR will also set this field to "FFFF FFFF" after every End of CS-PDU is processed.

Data drw 09

3.1.1.3 Free Buffer Descriptor (FBD)

The Free Buffer Descriptor (FBD) is a two (2) 32-bit word entry will contains the pointer for a receive free buffer. They are written into the local SRAM by the device driver. Its structure is shown as follows:

Free Buffer Descriptor (FBD)



Data drw 10

3.1.1.3.1 Free Buffer Descriptor (FBD): Word 1 of 2

FBD	BIT FIELDS
Word 1 of 2	31-0
	Buffer Handle

BIT FIELD	FUNCTION	DESCRIPTION
31-0	Buffer Handle	Buffer Handle. The Buffer Handle is used for virtual addressing which contains "virtual" start address of a free buffer. The device driver may call the operating system to allocate host memory and obtain the virtual start address. After a free buffer has been filled with cell payloads, the NICStAR writes this virtual start address, as part of the buffer status descriptor, into the Receive Status Queue. In "physical" memory environments, the Buffer Handle is the "physical" start address of a free buffer. Thus, in this case, the Buffer Handle is equal (initially) to the DMA Address. The device driver must write both the Buffer Handle and the DMA Address to one of the free buffer queues in the local SRAM using the Command Register.

Data drw 11

3.1.1.3.2 Free Buffer Descriptor (FBD): Word 2 of 2

Free Buffer Descriptor (FBD)

FBD	BIT FIELDS	
	31-2	1-0
Word 2 of 2	DMA Address	RSVD

BIT FIELD	FUNCTION	DESCRIPTION
31-2	DMA Address	DMA Address The DMA Address is the "physical" address of the next available location of the free buffer for storing information. Initially it will be loaded by the device driver as the starting pointer for the buffer. This value is updated by the NICStAR as cell payloads are stored into host memory. This value represents a 32-bit word boundary of the address since the two least significant bits (LSB) must be zero.
1-0	Zero	Must be zero.

Data drw 12

3.1.1.4 Receive Status Queue (RSQ)

The Receive Status Queue is a circular queue, located in host memory, which the NICStAR writes status information into after any host memory Free Buffer has been filled or an End of CS-PDU was detected. The device driver reads the queue to generate a list of host memory buffer addresses which constitute a reassembled CS-PDU, which it then passes to the application program(s) for converting the CS-PDU back to user data.

Each entry in the Receive Status Queue consists of 4 words, most of which is copied directly from the local SRAM's Receive Connection Table. The size of the queue is programmed at initialization in the Configuration Register (register #5); 128, 256 or 512 entries (2K, 4K or 8K bytes) and may not be changed during “run-time”. The host memory requirement for the RSQ is as follows:

Table 3: RSQ Size Requirement

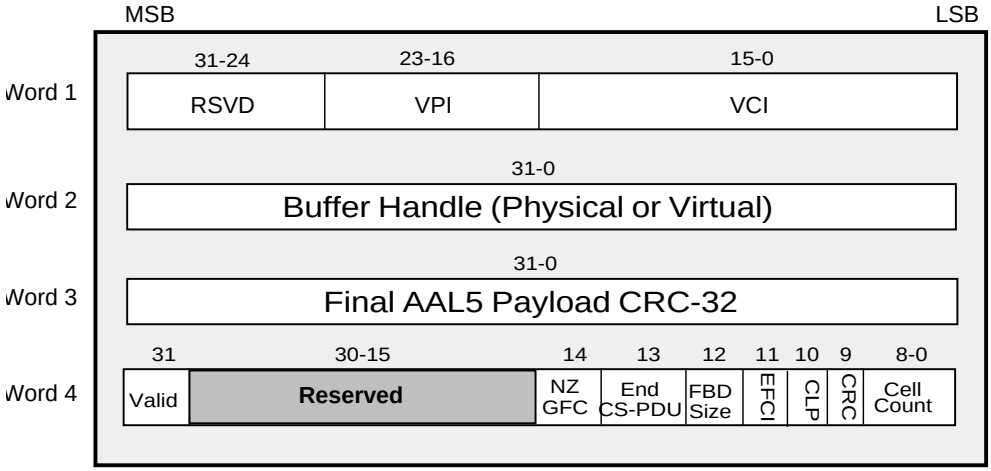
RSQ Size	Host Memory Required
128 entries	2048 bytes
256 entries	4096 bytes
512 entries	8192 bytes

At initialization, the device driver is responsible for allocating host memory for the queue and for zeroing every fourth word of the entry in the queue. The device driver then writes the base (start) address of the queue to the NICStAR's Receive Status Queue Base Register (register #7). As the NICStAR writes an entry into the queue, it will set bit 31 of word 4 to 1. In this manner, during “run-time,” the device driver only needs to initially read bit 31 of word 4 to determine if an entry is valid.

3.1.1.5 Receive Status Queue Entry (RSQE)

The following figures illustrate the Receive Status Queue Entries:

Entry in Receive Status Queue



Data drw 14

3.1.1.5.1 Receive Status Queue Entry (RSQE) Word 1 of 4

Receive Status Queue (RSQ) Entry

RSQE	BIT FIELDS		
	31-24	23-16	15-0
Word 1 of 4	RSVD	VPI	VCI

BIT FIELD	FUNCTION	DESCRIPTION
31-24	RSVD	Reserved. Set to 0 by the NICStAR.
23-16	VPI	VPI. This is the VPI field of the received ATM header. This bit field is written by the NICStAR.
15-0	VCI	VCI. This is the VCI field of the received ATM header. This bit field is written by the NICStAR.

Data drw 15

3.1.1.5.2 Receive Status Queue Entry (RSQE): Word 2 of 4

Receive Status Queue (RSQ) Entry

RSQE	BIT FIELDS
	31-0
Word 2 of 4	Buffer Handle

BIT FIELD	FUNCTION	DESCRIPTION
31-0	Buffer Handle	Buffer Handle. The Buffer Handle is the "virtual" start address of the buffer for the received cells. This is a copy of the Buffer Handle word written by the host to one of the free buffer queues in local SRAM. This bit field is written by the NICStAR.

Data drw 16

3.1.1.5.3 Receive Status Queue Entry (RSQE): Word 3 of 4

Receive Status Queue (RSQ) Entry

RSQE	BIT FIELDS
	31-0
Word 3 of 4	Final AAL5 CRC-32

BIT FIELD	FUNCTION	DESCRIPTION
31-0	Final AAL5 CRC-32	Final AAL5 Payload CRC-32. This bit field is used for AAL5 type only. Final value of the CRC-32 calculation for the AAL5 cell payloads of a CS-PDU. This is a copy of the current CRC-32 value contained within the Receive Connection Table of the current CS-PDU . If Word 4 bit 13 (End CS-PDU) is set, then it will become the final CRC-32 value of the current PDU. Normally this value is not needed by the host. However, in cases when the CRC-32 value calculated by the NICStAR does not equal the CRC-32 value contained within the last cell's payload (ie., the CRC Error flag is set to 1), this final CRC-32 value may be useful for diagnostic purposes.

Data drw 17

3.1.1.5.4 Receive Status Queue Entry (RSQE): Word 4 of 4

Receive Status Queue (RSQ) Entry

RCTE	BIT FIELDS								
	31	30-15	14	13	12	11	10	9	8-0
Word 4 of 4	Valid	**RSVD**	NZ GFC	End CS-PDU	FBD Size	EFCI	CLP	CRC	Cell Count

BIT FIELD	FUNCTION	DESCRIPTION
31	Valid	Valid Entry Tag. The NICStAR sets this bit to 1 to indicate a new entry into the Receive Status Queue. The device driver zeros this bit after it reads an entry (de-queues an entry). Thus, the device driver only needs to initially read this bit to determine if the entry is new (valid).
30-15	RSVD	Reserved. Set to zero.
14	NZ GFC	Non zero GFC. If set to 0, the GFC field was zero for all received ATM cells along this VC. If set to 1, one or more received ATM cells along this VC had a non-zero GFC field.
13	End CS-PDU	End of CS-PDU. If set to 1, the NICStAR detected an End of CS-PDU condition for either an AAL5 or "AAL0" format cell.(End of CS-PDU is detected in the PTI field for AAL5 and AAL0 formats and is detected in the ST field for AAL3/4). If equal to 0, the NICStAR did not detect an End of CS-PDU condition.

Data drw 18

Receive Status (RSQ) Entry, Word 4 of 4 - continued

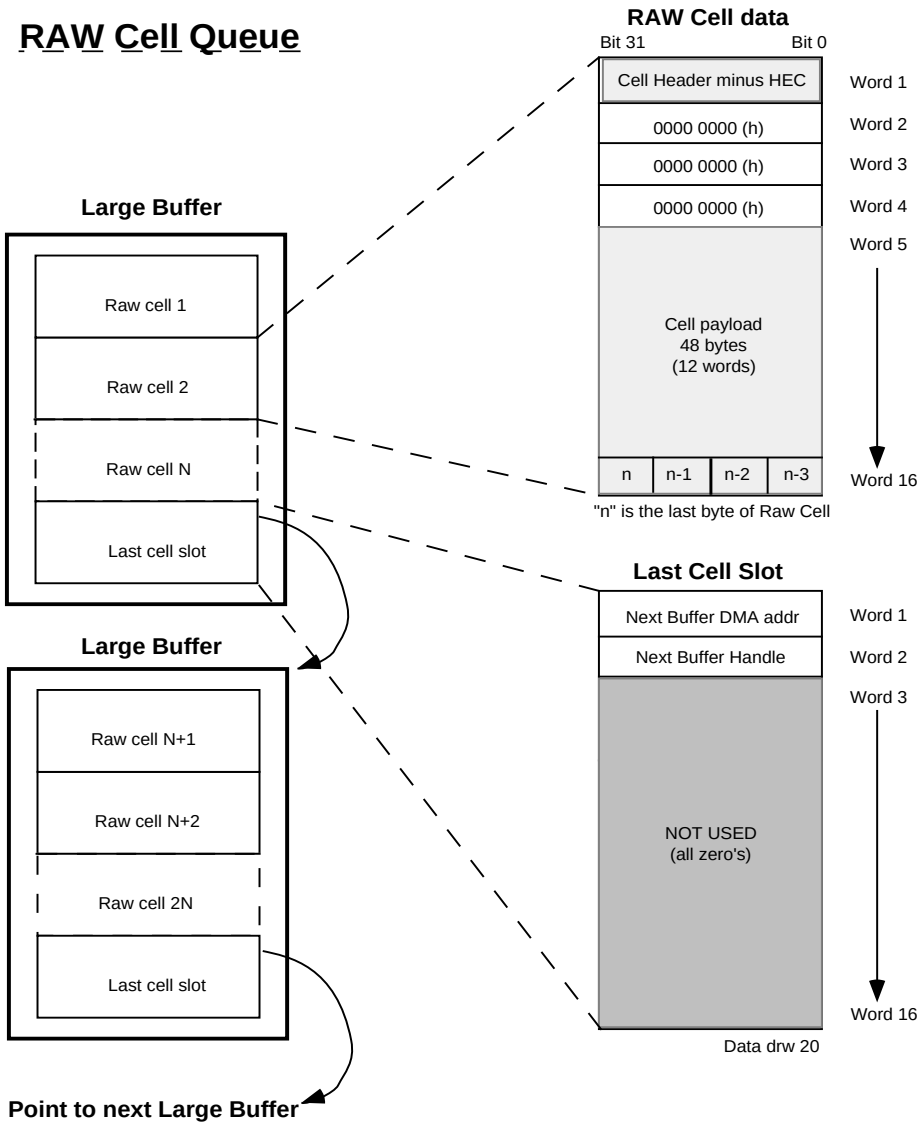
BIT FIELD	FUNCTION	DESCRIPTION
12	FBD Size	FBD Size. 0 = The current buffer was obtained from the Rx **Small** Free Buffer Queue. 1 = The current buffer was obtained from the Rx **Large** Free Buffer Queue.
11	EFCI	EFCI Congestion Flag. Default = 0. 0 = No congestion detected. 1 = Congestion detected. If the PTI field of the received ATM cell header is either PTI = 010 or = 011, then the NICStAR sets the congestion flag equal to one. After the complete CS-PDU has been reassembled, the NICStAR clears the flag to zero. Under normal operation, this bit equals to zero, no congestion detected.
10	CLP	Cell Loss Priority Flag. Default = 0. 0 = Received cell with CLP bit = 1 was **NOT** detected. 1 = Received cell with CLP bit = 1 was detected. If the CLP (Cell Loss Priority) bit of the received ATM cell header is set to one, then the CLP flag is set to one. After the complete CS-PDU has been reassembled, the NICStAR clears the flag to zero.
9	CRC	Received CRC Error. Default = 0. For AAL5 (CRC-32) and AAL3/4 (CRC-10) only. 0 = No CRC error detected at end of received CS-PDU. 1 = CRC error detected at end of received CS-PDU.
8-0	Cell Count	Cell Count. Specifies the number of ATM cell payloads which have been reassembled into the current buffer, where each cell payload represents 12 words. The maximum size of a Large Free Buffer is 16K bytes, thus the maximum number of cells per buffer is 341.

Data drw 19

3.1.1.6 Raw Cell Queue (RCQ)

The Raw Cell Queue (RCQ) consists of one or more Large Buffers in host memory. It is used by the NICStAR for storing some special ATM cells which should not be reassembled, such as flow control Resource Management (RM) cells, or OAM cells which contain important information in the cell header.

Data format for Raw Cell Queue in host memory is shown as follows:



3.1.2 Data Structure for Transmit Section

The data structure for the Transmit Section include the following:

Table 4: Location of Transmit Data Structures

Name of data structure	Location	Size (Bytes)
Transmit Schedule Table (TST)	local SRAM	4 - 40,816
TST Entry (TSTE)	local SRAM	4
Segmentation Channel Descriptor (SCD)	local SRAM	48
Segmentation Channel Queue (SCQ)	host memory	1K or 8K
Transmit Buffer Descriptor (TBD)	host memory	16
Transmit Status Request (TSR)	host memory	16
Transmit Status Indication (TSI)	host memory	8
Transmit Status Queue (TSQ)	host memory	8K

3.1.2.1 Transmit Schedule Table (TST)

The Transmit Schedule Table (TST) is located in the local SRAM which is used to tell the NICStAR “where from”, “when to” and “transmit a cell”. Each entry in the TST is one 32-bit word and that represent one cell slot time. The NICStAR scans the TST entries on a periodic basis, one entry per cell time slot. The last entry in the TST is used as jump address and it doesn't take up a cell time slot. The jump address contains the address of the local SRAM location for the NICStAR to jump to, and it will continue sequentially until it hits another branch location in the new TST.

3.1.2.2 Transmit Schedule Table Entry (TSTE)

The entry of the TSTE is shown as follows:

Transmit Schedule Table Entry (TSTE)

TSTE	BIT FIELDS			
	31	30-29	28-17	16-0
Word 1 of 1		TSTOP	RSVD	SRAMAD

BIT FIELD	FUNCTION	DESCRIPTION
31	----	Reserved. Set to zero.
30-29	TSTOP	TST Opcode. This Opcode. specifies the action the NICStAR will perform for the TST entry. 00 = Insert a forced Null cell into the Tx FIFO; SRAMAD field is not used. 01 = Transmit fixed rate cell; SRAMAD field is valid. 10 = Opportunity to transmit an variable rate cell;SRAMAD field is not used. 11 = End_of_TST branch; SRAMAD field is valid.
28-17	RSVD	Reserved. Set to zero.
16-0	SRAMAD	SRAM Address. When TSTOP = 01, this field specifies the SRAM address of an SCD for that fixed rate channel. When TSTOP = 11, this field specifies the SRAM address of the next location of the TST entry. When TSTOP = 00 or 10, this field is not used.

Data drw 22

3.1.2.3 Segmentation Channel Descriptor (SCD)

The Segmentation Channel Descriptor (SCD) is a twelve word data structure located in the local SRAM. For each SCQ located in host memory, there is an associated SCD located in local SRAM. The first four words of an SCD contain information about its SCQ Base, Tail, and Head addresses. The Base address is the beginning of the SCQ. The head pointer is the last entry in the SCQ inserted by the device driver.

Since each SCQ requires a SCD which tells the NICStAR where the SCQ is located in the host memory, the device driver has to initialize all the SCQ's by writing the first four (4) words for each SCD in the SRAM.

The next eight (8) words in the SCD contain two cached entries (4 words each) of the TBD/TSR's in the SCQ. The NICStAR is responsible for fetching TBD/TSR into these two cached entries in the local SRAM. These two cached entries are served as the working entries for the NICStAR during run-time operation.

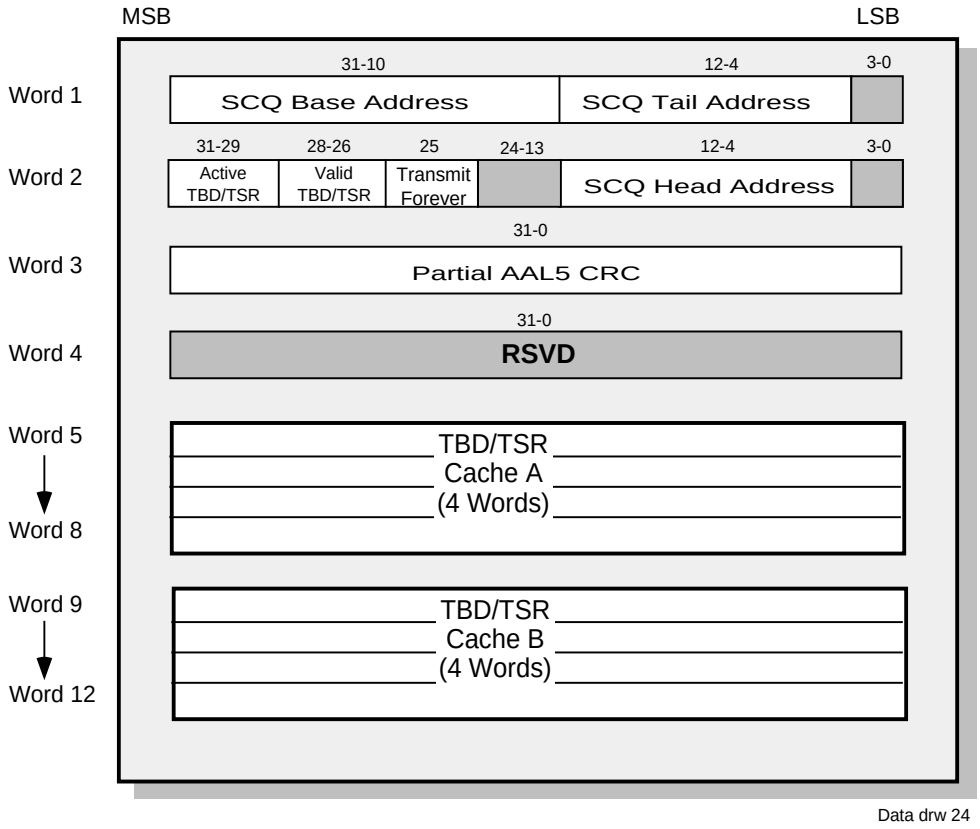
The SCD's for VBR SCQ's are VBR SCD0, VBR SCD1 and VBR SCD2. These three SCD's located in fixed addresses in the local SRAM as follows:

Table 5: VBR SCD Locations

SCD Name	SRAM Address	PRIORITY
VBR SCD2	1 E7DC - 1 E7E7	Low
VBR SCD1	1 E7E8 - 1 E7F3	Mid
VBR SCD0	1 E7F4 - 1 E7FF	High

The format of the Segmentation Channel Descriptor (SCD) is shown below:

Segmentation Channel Descriptor (SCD)



Note: The first word is routinely updated by the Host Drive and is the only word that must be modified by the Driver during renting. The remaining words are described for diagnostic purposes only.

3.1.2.3.1 Segmentation Channel Descriptor (SCD): Word 1 of 12

Segmentation Channel Descriptor (SCD)

SCD	BIT FIELDS		
	31-10	12-4	3-0
Word 1 of 12	SCQ Base Addr	SCQ Tail Addr	RSVD

BIT FIELD	FUNCTION	DESCRIPTION
31-10	SCQ Base Addr	SCQ Base Address. Specifies the SCQ Base Address in host memory. Size of this bit field depends on type of channels as follows: Channel Type — SCQ Base — SCQ Tail fixed rate — 31:10 — 9:4 variable rate — 31:13 — 12:4 This field is written by the host and read by the NICStAR.
12-4	SCD Tail Addr	SCQ Tail Address. The SCQ Tail address is used with the SCQ Base address to form a complete address pointing to the host memory which specifies the last SCQ entry for the NICStAR to process. Channel Type — SCQ Base — SCQ Tail fixed rate — 31:10 — 9:4 variable rate — 31:13 — 12:4 This field is **written by the host Drive** and read by the NICStAR. Address 3:0 must be zero, since all SCQ entry are 4 word long (16 bytes)
3-0	RSVD	Reserved. Must be all zero's.

Data drw 25

Note: This word is routinely updated by the Host Drive and is the only word that must be modified by the Driver during renting. The remaining words are described for diagnostic purposes only.

3.1.2.3.2 Segmentation Channel Descriptor (SCD): Word 2 of 12

Segmentation Channel Descriptor (SCD)

SCD	BIT FIELDS					
	31-29	28-26	25	24-13	12-4	3-0
Word 2 of 12	Active TBD/TSR	Valid TBD/TSR	Transmit Forever	**RSVD**	SCQ Head Addr	**RSVD**

BIT FIELD	FUNCTION	DESCRIPTION
31-29	Active TBD/TSR	Active TBD/TSR. This bit field describes the active TBD/TSR currently used by the NICStAR whether it is in cache A or cache B of the SCD. This bit field normally is not required to be read by the device driver. It is included here for **diagnostic purpose only**. 31 30 29 — Description 0 0 0 — Active TBD/TSR is NOT A and NOT B. 0 0 1 — Active TBD/TSR is A. 0 1 0 — Active TBD/TSR is B. 0 1 1 — Reserved. 1 0 0 — Reserved. 1 0 1 — Reserved. 1 1 0 — Reserved. 1 1 1 — Reserved. This field is initialized to zero by the host and then write/read by the NICStAR.
28-26	Valid TBD/TSR	Valid TBD/TSR. This bit field describes whether the TBD/TSR's currently in cache A or cache B of the SCD is valid or not. This bit field normally is not required to be read by the device driver. It is included here for **diagnostic purpose only**. 28 27 26 — Description 0 0 0 — Valid TBD/TSR is NOT A and NOT B. 0 0 1 — Valid TBD/TSR is A. 0 1 0 — Valid TBD/TSR is B. 0 1 1 — Valid TBD/TSR is A and B. 1 0 0 — Reserved. 1 0 1 — Reserved. 1 1 0 — Reserved. 1 1 1 — Reserved. This field is initialized to zero by the host and then write/read by the NICStAR.

Data drw 26

Segmentation Channel Descriptor (SCD), Word 2 of 12 - continued

BIT FIELD	FUNCTION	DESCRIPTION
25	Transmit Forever	Transmit Forever. 0 = Normal operation. 1 = Transmit Forever. If this bit is set, the NICStAR will segment cells from the current buffer and it will fetch the same TBD over and over again. If a TSR is being accessed, a TSI will be repeatedly generated. In both cases, both caches in the SCD will be used.. In this mode, the transmit buffer must be a multiple of 12 words. The host writes to this field **only** when the SCQ Head and Tail values are **equal**.
24-13	RSVD	Reserved. Must be zero's.
12-4	SCQ Head Addr	SCQ Head Address. The SCQ Head address is concatenated with the SCQ Base address to form a complete address pointing to the host memory which specifies the next SCQ entry for the NICSTAR to process. Channel Type / SCQ Base / SCQ Head CBR / 31:10 / 9:4 non-CBR / 31:13 / 12:4 Address 3:0 must be zero, since all SCQ entry are 4 word long (16 bytes)
3-0	RSVD	Reserved. Must be zero's.

Data drw 27

3.1.2.3.3 Segmentation Channel Descriptor (SCD): Word 3 and 4 of 12

Segmentation Channel Descriptor (SCD)

SCD	BIT FIELDS
	31-0
Word 3 of 12	Partial AAL5 CRC-32

BIT FIELD	FUNCTION	DESCRIPTION
31-0	Partial AAL5 CRC-32	Partial AAL5 CRC-32. This is a write/read location used by the NICStAR to store current "running total" of the CRC-32 calculation for the AAL5 cell payloads of a CS-PDU. This word must be set to "FFFFFFFF" during initialization by the host. Value is ignored for non-AAL5 type.

Segmentation Channel Descriptor (SCD)

SCD	BIT FIELDS
	31-0
Word 4 of 12	RSVD

BIT FIELD	FUNCTION	DESCRIPTION
31-0	RSVD	Reserved. Must be set all zero's by the host.

Data drw 28

3.1.2.3.4 Segmentation Channel Descriptor (SCD): Word 5 to 8 of 12

Segmentation Channel Descriptor (SCD)

SCD	BIT FIELDS
	31-0
Word 5 of 12	(Cache A) TBD/TSR Word 1

SCD	BIT FIELDS
	31-0
Word 6 of 12	(Cache A) TBD/TSR Word 2

SCD	BIT FIELDS
	31-0
Word 7 of 12	(Cache A) TBD/TSR Word 3

SCD	BIT FIELDS
	31-0
Word 8 of 12	(Cache A) TBD/TSR Word 4

BIT FIELD	FUNCTION	DESCRIPTION
31-0	Cache A TBD/TSR	Cache A TBD/TSR. These four words contains the current working copy of the TBD/TSR fetched by the NICStAR from the SCQ. See data structure on TBD/TSR for detailed description. These four words should be initialized by the host to zeros at start up.

Data drw 29

3.1.2.3.5 Segmentation Channel Descriptor (SCD): Word 9 to 12 of 12

Segmentation Channel Descriptor (SCD)

SCD	BIT FIELDS
Word 9 of 12	31-0
	(Cache B) TBD/TSR Word 1

SCD	BIT FIELDS
Word 10 of 12	31-0
	(Cache B) TBD/TSR Word 2

SCD	BIT FIELDS
Word 11 of 12	31-0
	(Cache B) TBD/TSR Word 3

SCD	BIT FIELDS
Word 12 of 12	31-0
	(Cache B) TBD/TSR Word 4

BIT FIELD	FUNCTION	DESCRIPTION
31-0	Cache B TBD/TSR	Cache B TBD/TSR. These four words contains the current working copy of the TBD/TSR fetched by the NICStAR from the SCQ. See data structure on TBD/TSR for detailed description. These four words should be initialized by the host to zeros at start up.

Data drw 30

3.1.2.4 Segmentation Channel Queue (SCQ)

The SCQ is created by the device driver for controlling the transmit section of the NICStAR. The SCQ is composed of TBD's and/or TSR's in the host memory. The NICStAR will fetch each TBD/TSR in the SCQ one after another in sequence. The device driver can grow the size of a SCQ by adding new entries (TBD's or TSR's) and move the SCQ tail pointer forward to reflect the new size. The NICStAR processes the SCQ and moves the head pointer to indicate its currently servicing entry.

There is one SCQ for each fixed rate VC. However, there are only three (3) SCD's for all the variable rate channels (i.e. VBR/VBR VC's). These three variable rate SCQ's are priority based. They are named VBR SCQ0, VBR SCQ1 and VBR SCQ2. VBR SCQ0 is highest priority, VBR SCQ1 is second highest while VBR SCQ2 is the lowest.

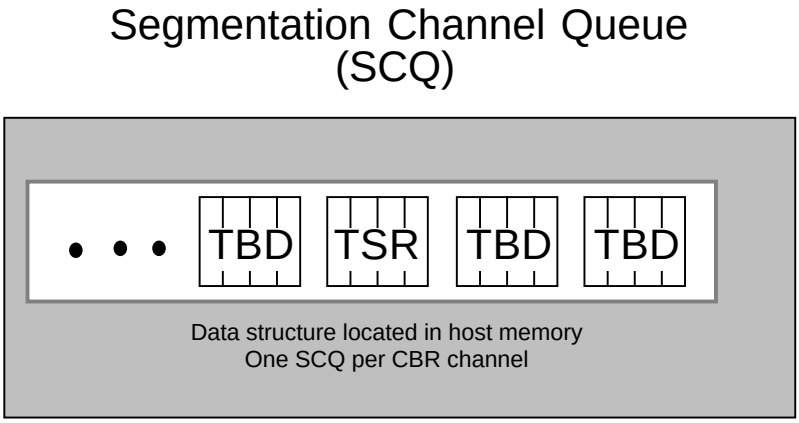
There is a maximum number of TBD/TSR's for each type of SCQ based on the SCQ type. They are shown as follows:

Table 6: SCQ Size for various VC types

VC Type	Max. TBD/TSR (4 Dwords or 16 Bytes each)	Host Memory Required	Empty Slot
fixed rate	63	1024 bytes	1
variable rate	511	8192 bytes	1

The host can advance the pointer for the fixed rate channels max. 63 entries at one time, even though storage is for 64 entries. Also, the host can only advance the pointer for the variable rate channels 511 entries, even though storage is for 512 entries.

The data structure of SCQ is shown below:



3.1.2.5 Transmit Buffer Descriptor (TBD)

Transmit Buffer Descriptor (TBD) is a four 32-bit word entry located in the host memory which is used to describe a transmit buffer. A CS_PDU can be broken down into multiple TBD's and chained together in a queue called Segmentation Channel Queue (SCQ). In a typical CS_PDU containing multiple TBD's, the finished data structure for the SCQ will be a list of consecutive four (4) 32-bit word entries for all the TBD's. The order of transmission will follow the order of each TBD in the SCQ. The NICStAR will be told through the Segmentation Channel Descriptor (SCD) which contain information such as the base (beginning) of the SCQ in host memory.

When building TBD's, the following rules must be adhered to:

1) The ***Buffer Length*** field **must not** be zero.

2) The ***Buffer Length*** field of an individual TBD must be a multiple of four bytes. This limitation can be removed with the addition of a PAL device allowing MOD 1 capabilities. This PAL device is explained in the appendix.

3) The sum of the ***Buffer Length*** fields of one or more TBD(s) describing a CS-PDU must be a multiple of 48 bytes (the device driver must pad data to be a multiple of 48 bytes).

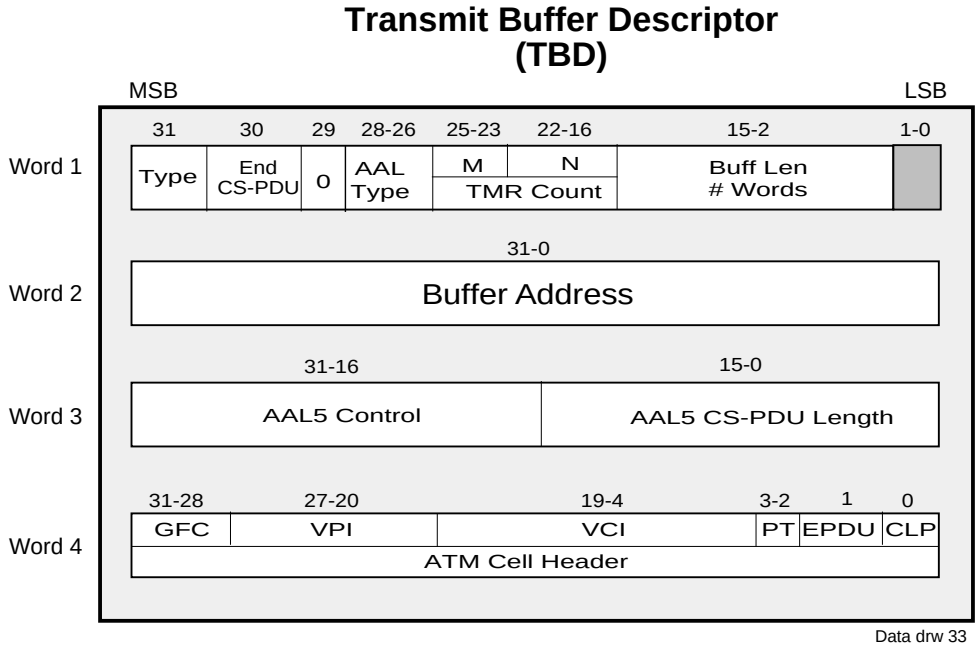
4) When using multiple TBDs to describe a single CS-PDU, the TBDs must be contiguous and grouped (including no interleaved TSRs).

5) For AAL5 only, when calculating the ***Buffer Length*** field for a TBD(s) describing a CS-PDU the device driver must include the length needed to hold the ***AAL5 length/control*** field and ***AAL5 CRC-32***.

6) For AAL5 only, when using multiple TBDs to describe a single AAL5 CS-PDU, the ***Buffer Length*** field of the last TBD for the CS-PDU must be greater than 8-bytes.

7) For AAL5 only, when the last cell of the CS-PDU spans across multiple TBD's, all TBD's contains part of the last cell must have the ***End_CS_PDU*** bit set to "1".

The following is the format of the Transmit Buffer Descriptor (TBD):



3.1.2.5.1 Transmit Buffer Descriptor (TBD): Word 1 of 4

Transmit Buffer Descriptor (TBD)

TBD	BIT FIELDS							
	31	30	29	28-26	25-23	22-16	15-2	1-0
Word 1 of 4	TYPE	End CS-PDU	RSVD	AAL Type	M TMR Count	N	Buff Len	0 0

BIT FIELD	FUNCTION	DESCRIPTION
31	TYPE	Descriptor type. Set to 0, specifies the descriptor is a TBD.
30	End CS-PDU	End of CS-PDU. If set to 1, specifies that the corresponding Tx Buffer is the last buffer of an AAL5 CS-PDU. In this case, the NICStAR will substitute its CRC-32 calculation for the CRC-32 "placeholder" in the last four bytes of the buffer. If equal to 0, the corresponding buffer is the first or an intermediate Tx Buffer of an AAL5 CS-PDU. This bit field applies to AAL5 and "AAL0" formats only.
29	RSVD	Reserved. Set to 0.
28-26	AAL Type	AAL Type. Specifies the AAL format of the information in the Tx Buffer; 000 = AAL0; 001 = AAL3/4 ; 010 = AAL5 011 = Not used; 100 = Not used; 101 = Not used; 110 = Not used; 111 = Not used.

Data drw 34

Transmit Buffer Descriptor (TBD), Word 1 of 4 - continued

BIT FIELD	FUNCTION	DESCRIPTION
25-23	M	M Value Timer Count For fixed rate channel, set to 000. For OAM and variable rate channel (VBR) only: VBR Rate Counter M. This bit field specifies the numerator scaling factor used for transmitting OAM, VBR or UBR data. This value is used together with the VBR Rate Counter N bit field below (e.g., for 155 ATM, gross bit rate = (M/N)*(155.52)). (for additional information, please refer to the VBR SCD/SC Queues section)
22-16	N	N Value Timer Count For CBR channel, set to 000. For OAM, VBR and UBR only: VBR Rate Counter N. This bit field specifies the numerator scaling factor used for transmitting OAM, VBR or UBR data. This value is used together with the VBR Rate Counter M bit field below (e.g., for 155 ATM, gross bit rate = (M/N)*(155.52)). (for additional information, please refer to the VBR SCDs/SC Queues section)
15-2	Buff Len	Buffer Length. Specifies the size of a Tx Buffer in words, which makes it modulo 4 (a multiple of 4) byte size. The buffer length can be up to 16K-1 words (or 64K-4 bytes). These bits can be used together with bits 1 and 0 below in byte format. Mod 4 applies to Rev D only, mod 48 applies to both Rev. C3 and Rev. D. When build a CS-PDU for transmit, the following rules must be observed: i) Buffer Length must be in word boundary (4-byte) ii) Buffer Length must NOT be zero iii) the sum of Buffer Length fields of all TBD's belong to the same CS-PDU must be a multiple of 48 bytes.
1-0	Zero	Must be set to 0. If a PAL is added to the PCI interface (see the PAL equation and discription in the appendix), this field can be programmed to any value. This will allow any MOD 1 Buffer size to be defined.

Data drw 35

3.1.2.5.2 Transmit Buffer Descriptor (TBD): Word 2 of 4

Transmit Buffer Descriptor (TBD)

TBD	BIT FIELDS
Word 2 of 4	31-0
	Buffer Address

BIT FIELD	FUNCTION	DESCRIPTION
31-0	Buffer Address	Buffer Address. The starting address of a host memory Tx Buffer. This address may begin on any byte boundary. The address must exist in the physical PCI address space.

Data drw 36

3.1.2.5.3 Transmit Buffer Descriptor (TBD): Word 3 of 4

Transmit Buffer Descriptor (TBD)

TBD	BIT FIELDS	
Word 3 of 4	31-16	15-0
	AAL5 Control	AAL5 CS-PDU Length

BIT FIELD	FUNCTION	DESCRIPTION
31-16	AAL5 Control	AAL5 Control. This bit field will be loaded into bit positions 31-16 of word 11 of the last AAL5 cell payload. This Control bit field is currently undefined by the ATM Forum and it should be set to 0. This bit field is reserved if AAL5 is not used.
15-0	AAL5 CS-PDU Length	AAL5 CS-PDU Length. Specifies the length of the AAL5 CS-PDU in bytes. This bit field will be loaded into bit positions 15-0 of word 11 of the last AAL5 cell payload. This bit field is reserved if AAL5 is not used.

Data drw 37

3.1.2.5.4 Transmit Buffer Descriptor (TBD): Word 4 of 4

Transmit Buffer Descriptor (TBD)

TBD	BIT FIELDS					
Word 4 of 4	31-28	27-20	19-4	3-2	1	0
	GFC	VPI	VCI	PT	EPDU	CLP
	ATM Cell Header					

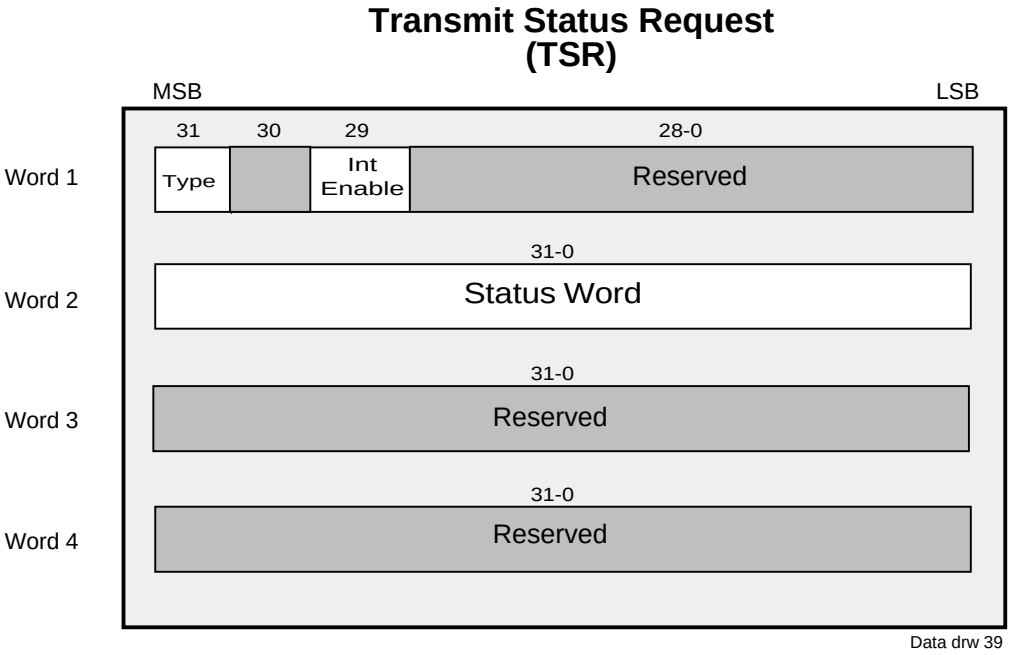
BIT FIELD	FUNCTION	DESCRIPTION
31-28	GFC	GFC. This is the GFC field of the transmitting ATM header.
27-20	VPI	VPI. This is the VPI field of the transmitting ATM header.
19-4	VCI	VCI. This is the VCI field of the transmitting ATM header.
3-2	PT	PT. This is the PT field of the transmitting ATM header. If the CS-PDU is AAL5 or AAL0, then the NICStAR will modify this field for the last Tx cell to indicate end of PDU.
1	EPDU	End of PDU Bit. This bit is generated from Bit 30 of word one (End CS-PDU bit in the TBD Register).
0	CLP	CLP. This is the CLP field of the transmitting ATM header.

Data drw 38

3.1.2.6 Transmit Status Request (TSR)

In addition to the TBD's in the Segmentation Channel Queue (SCQ), there is another data structure called Transmit Status Request (TSR). A TSR can be inserted in the SCQ to cause the NICStAR to generate a Transmit Status Indicator (TSI) in the Transmit Status Queue (TSQ). By using TSRs, the device driver can track the NICStAR's transmit progress.

The following is the format of the Transmit Status Request (TSR):



3.1.2.6.1 Transmit Status Request (TSR): Word 1 of 4

Transmit Status Request (TSR)

TSR	BIT FIELDS			
	31	30	29	28-0
Word 1 of 4	Type	RSVD	Int Enable	RSVD

BIT FIELD	FUNCTION	DESCRIPTION
31	Type	Descriptor Type. Set to 1, specify the descriptor is a TSR, Transmit Status Request. (set to 0, for TBD, Transmit Buffer Descriptor)
30	RSVD	Reserved. Set to 0.
29	Int Enable	Interrupt Enable. If set to 1, the NICStAR will generate an interrupt to the host system after it writes the Transmit Status Indicator (TSI) into the Transmit Status Queue. If equal to 0, the NICStAR will not generate an interrupt to the host system after it writes the TSI into the Transmit Status Queue.
28-0	RSVD	Reserved. Set to 0.

Data drw 40

3.1.2.6.2 Transmit Status Request (TSR): Word 2 of 4

Transmit Status Request (TSR)

TSR	BIT FIELDS
	31-0
Word 2 of 4	Status Word

BIT FIELD	FUNCTION	DESCRIPTION
31-0	Status	Status. The Status Word could be any value and it is created by the device driver. This status word will be written (unmodified) by the NICStAR into the Transmit Status Indicator (TSI) in host memory when the NICStAR processes the TSR.

Data drw 41

3.1.2.6.3 Transmit Status Request (TSR): Word 3 & 4 of 4

Transmit Status Request (TSR)

TSR	BIT FIELDS
	31-0
Word 3 of 4	Reserved

Transmit Status Request (TSR)

TSR	BIT FIELDS
	31-0
Word 4 of 4	Reserved

BIT FIELD	FUNCTION	DESCRIPTION
31-0	Reserved	Reserved. Set to 0.

Data drw 42

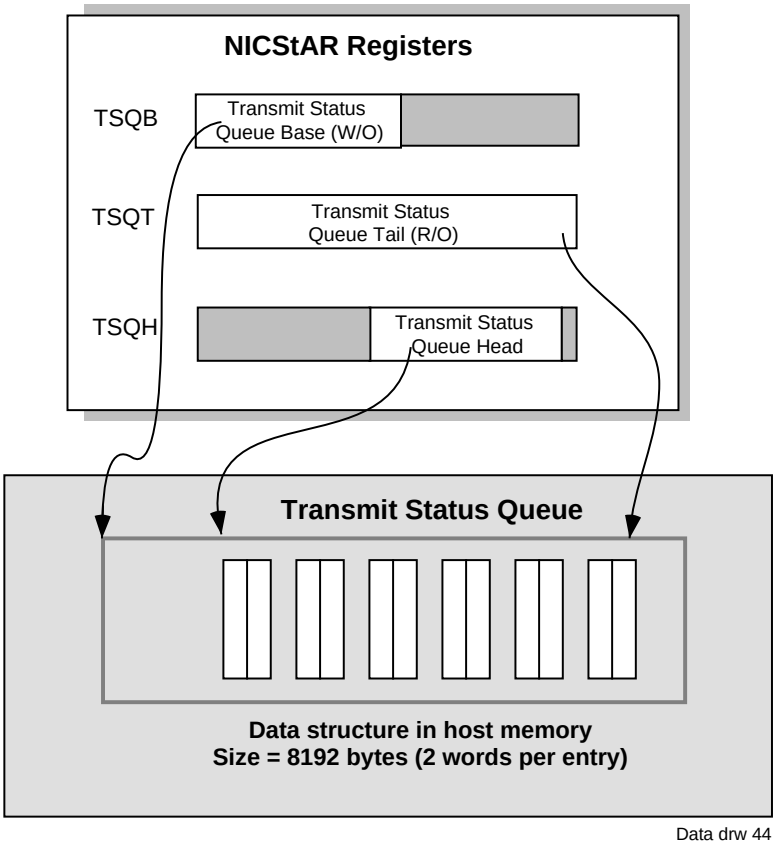
3.1.2.7 Transmit Status Queue (TSQ)

The Transmit Status Queue (TSQ) is located in the host memory. This is the place where the NICStAR will put the Transmit Status Indicators (TSI). The size of this queue is fixed 1024 entries which requires 8192 bytes of host memory.

Table 7: TSQ Size Requirement

TSQ Size	Host Memory Required
1024 entries	8192 bytes

This data structure is specified by three registers inside the NICStAR as follows:



Data drw 44

Transmit Status Queue Base - specify base address of the TSQ in the host memory.

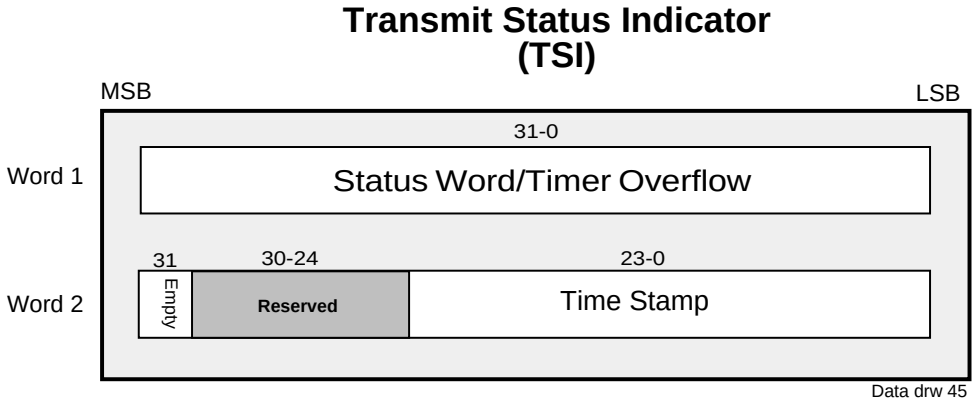
Transmit Status Queue Tail - specify address of next entry in the TSQ for the NICStAR to write a TSI

Transmit Status Queue Head - specify address of next entry in the TSQ for the host to read a TSI

3.1.2.8 Transmit Status Indicator (TSI)

When the NICStAR encounters a Transmit Status Request (TSR) in the SCQ, it will generate a Transmit Status Indicator (TSI) and put it in the Transmit Status Queue (TSQ).

The TSI is a two word data structure as shown as follows:



Data drw 45

3.1.2.8.1 Transmit Status Indicator (TSI): Word 1 of 2

Transmit Status Indicator (TSI)

TSI	BIT FIELDS
	31-0
Word 1 of 2	Status /Timer Overflow

BIT FIELD	FUNCTION	DESCRIPTION
31-0	Status/ Timer Overflow	Status/Timer Overflow. If the Timer (TMR) Register rolls over, this word will be written by the NICStAR with a value zero. If the Timer (TMR) Register did not roll over, the NICStAR will copy (write) the status word field from the Transmit Status Request (TSR) when a Transmit Status Request (TSR) is processed in the Segmentation Channel Queue (SCQ).

Data drw 46

3.1.2.8.2 Transmit Status Indicator (TSI): Word 2 of 2

Transmit Status Indicator (TSI)

TSI	BIT FIELDS		
	31	30-24	23-0
Word 2 of 2	Empty	RSVD	Time Stamp

BIT FIELD	FUNCTION	DESCRIPTION
31	Empty	Empty. 0 = Entry is valid. 1 = Entry is not valid. This bit should be set to "1" by the device driver as part of the initialization of the Transmit Status Queue. The NICStAR will reset this bit to "0" to indicate the host that this entry is valid as a result of the Timer Register roll over or a Transmit Status Request (TSR) in the Segmentation Channel Queue (SCQ). The host should set this bit back to "1" after servicing the valid entry.
30-24	RSVD	Reserved. Set to zero.
23-0	Time Stamp	Time Stamp. If the Timer (TMR) Register rolls over, this field will be written with a value zero. If the Timer (TMR) Register did not roll over, the NICStAR will copy the current value of the Timer Register into this field when a Transmit Status Request (TSR) is processed in the Segmentation Channel Queue (SCQ).

Data drw 47

3.2 Local SRAM Memory Maps

The SRAM memory space for the NICStAR is 00000(h) - 1FFFF(h). All accesses to the local SRAM are 32-bit word only to provide a total address space of 128K x 32. The NICStAR supports two SRAM configurations: 128K x32 and 32K x 32. The device driver accesses the local SRAM indirectly through the Command and Data Registers (refer to NICStAR Registers section for details). When 128K x 32 SRAM is installed, the whole memory space is populated. When 32K x 32 SRAM is installed, the memory space 04000(h) - 1BFFF(h) is not used (total 96K x 32).

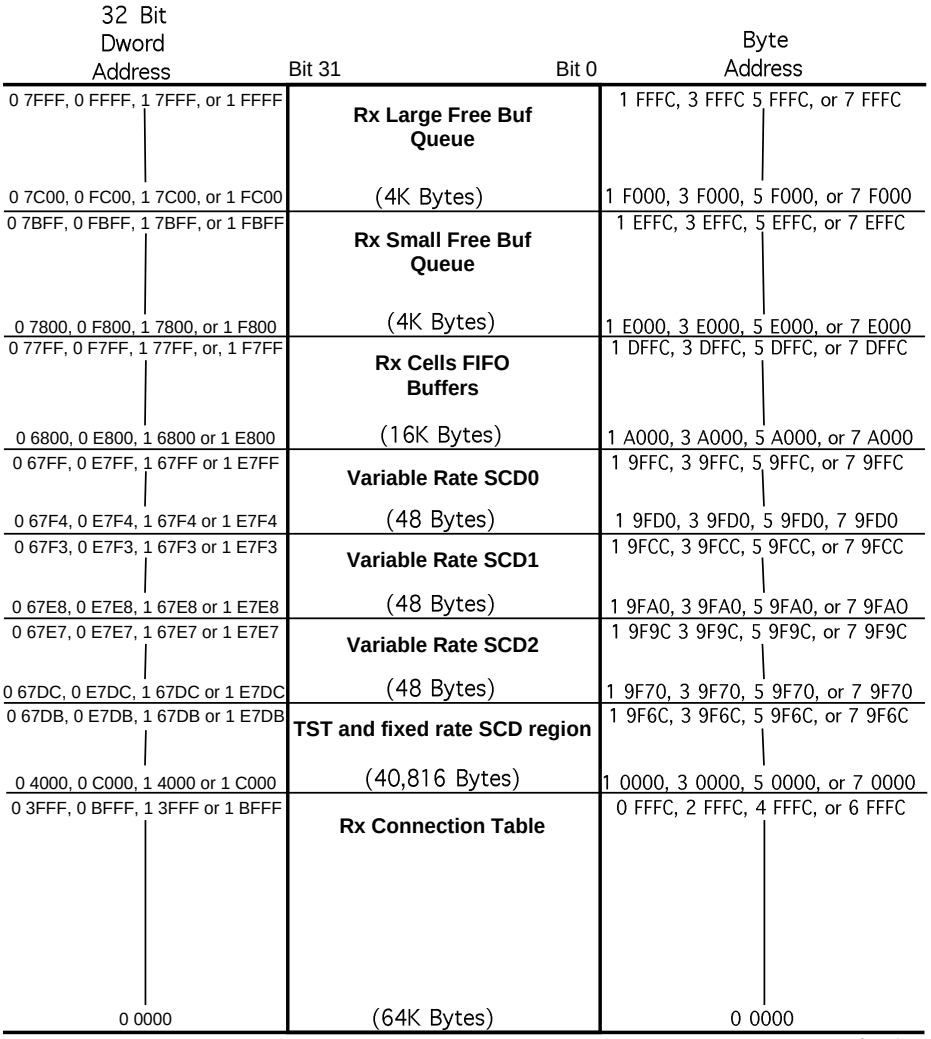
Table 8: Functional Components in two SRAM Configurations

Functional Components	32K Dword SRAM Location (Bytes)	128K Dword SRAM Location (Bytes)	Entry Size (Bytes)
Rx Large Free Buffer Queue	1FC00 - 1FFFF (4K)	1FC00 - 1FFFF (4K)	8
Rx Small Free Buffer Queue	1F800 - 1FBFF (4K)	1F800 - 1FBFF (4K)	8
Rx Cell FIFO Buffers	1E800 - 1F7FF (16K)	1E800 - 1F7FF (16K)	25/Cell
VBR SCD0	1E7F4 - 1E7FF (48)	1E7F4 - 1E7FF (48)	48
VBR SCD1	1E7E8 - 1E7F3 (48)	1E7E8 - 1E7F3 (48)	48
VBR SCD2	1E7DC - 1E7E7 (48)	1E7DC - 1E7E7 (48)	48
TST and CBR SCD's	1C000 - 1E7DB (40,816)	1C000 - 1E7DB (40,816)	TBD: 4 SCD: 12
Not Used	N/A	10000 - 1BFFF (192K)	N/A
Rx Connection Table	0 0000 - 1 BFFF (64K)	00000 - 0FFFF (265K)	16

where “K” = 1024, and a D word = 32 bits.

It is to be noted that the Rx Cell FIFO Buffer in the local SRAM does not need to be accessed by the device driver. The address is provided here for reference only.

NICStAR SRAM Memory Map For Using 32k X 32 SRAM

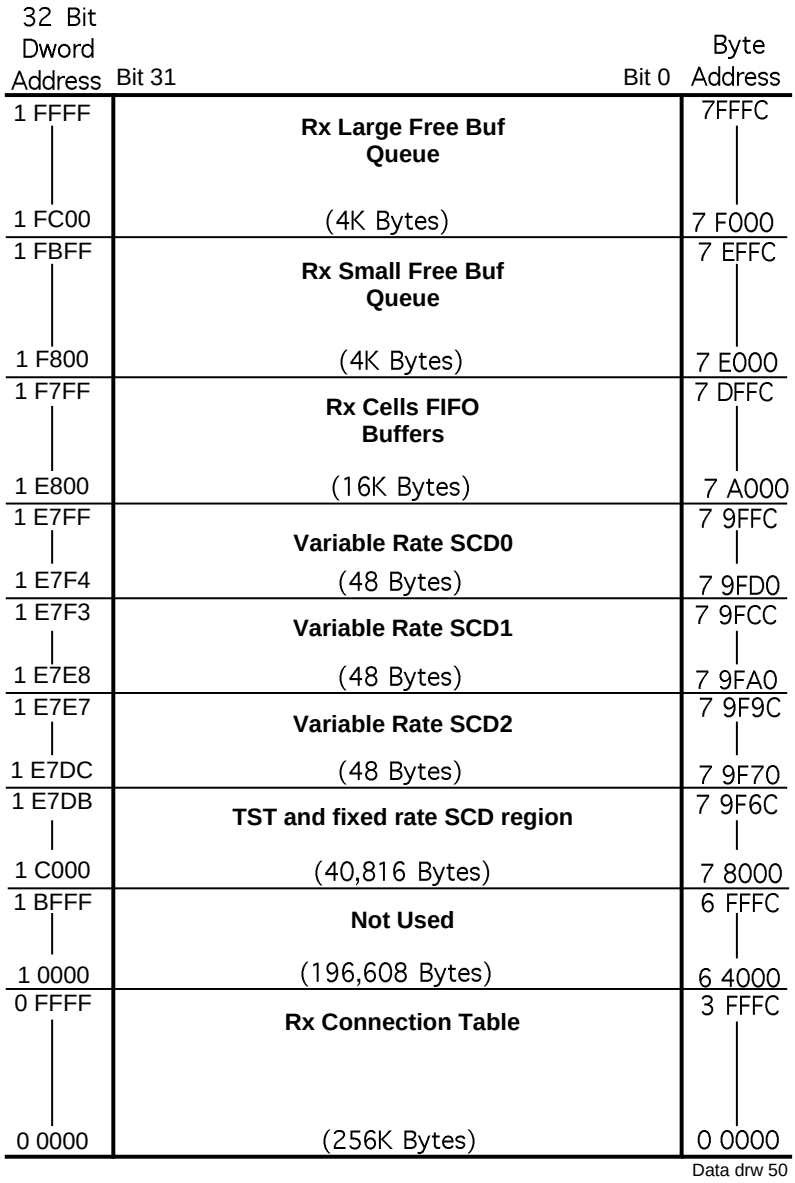


Data drw 49

Note: The 32K X 32 memory array is mapped into the 128K X 32 memory array so that a driver could access either of them by using the highest (128K X 32) addresses (17-bit address). The 32K X 32 memory array is only a 15-bit address so the two highest order address bits are not used. This map shows the aliasing.

32-bit Double word addresses are the SRAM addresses - the byte address is the address written to the command register shifted left by two bits (x4) including the two least significant reserved bits.

NICStAR SRAM Memory Map For Using 128k X 32 SRAM



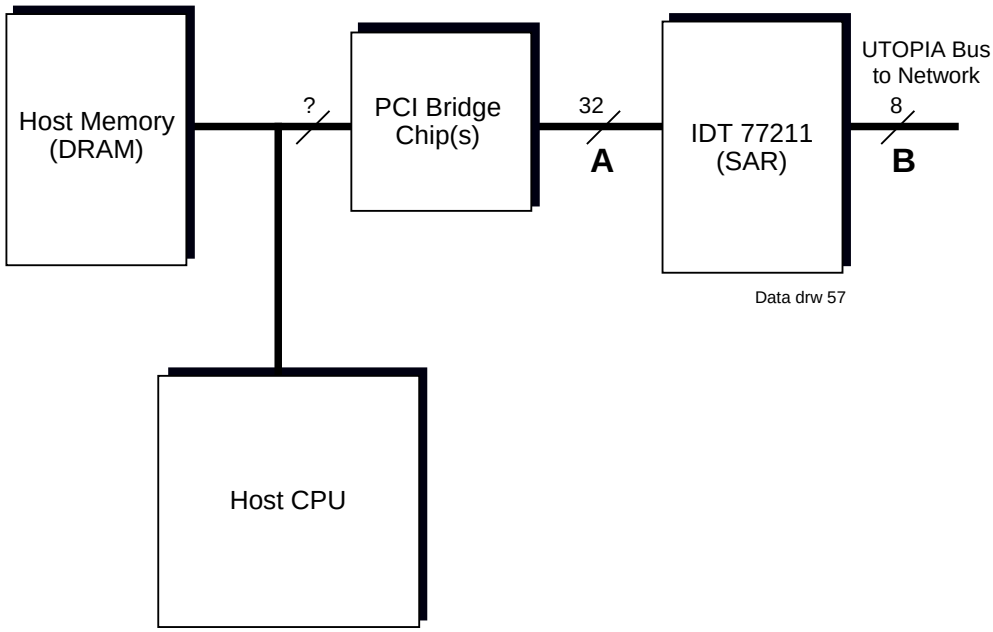
Data drw 50

Note: 32-bit Double word addresses are the SRAM addresses - the byte address is the address written to the command register shifted left by two bits (x4) including the two least significant reserved bits.

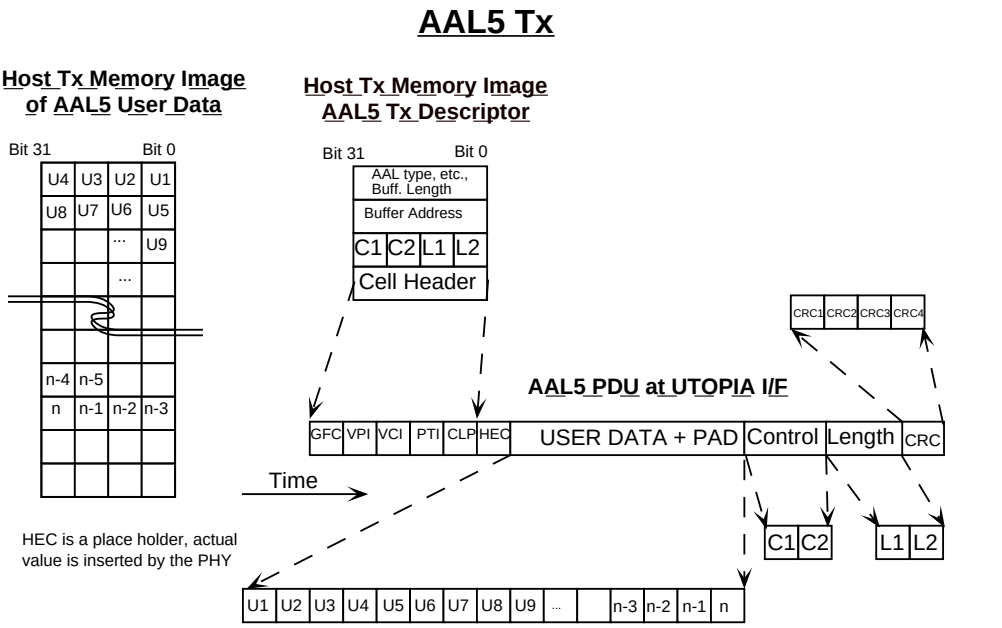
3.3 AAL Data Formats in Host Memory

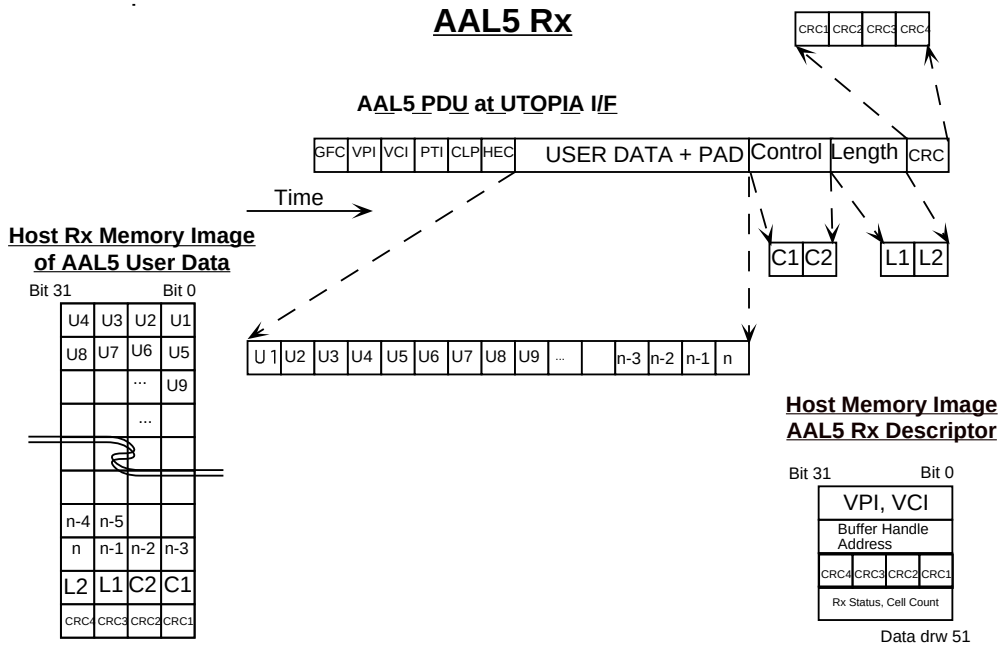
This section describes the data formats for AAL0, AAL3/4, and AAL5 when they are received/transmitted from/to the UTOPIA interface into and out of the host memory. The 77211 supports little endian as well as big endian operation.

The following pages describe data formats at points "A" and "B" in the system.

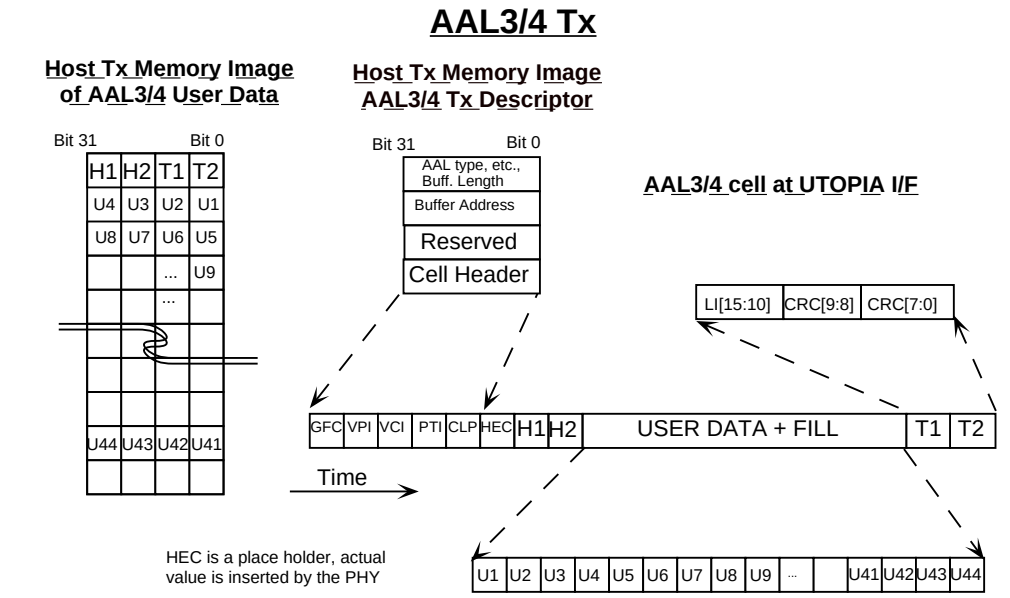


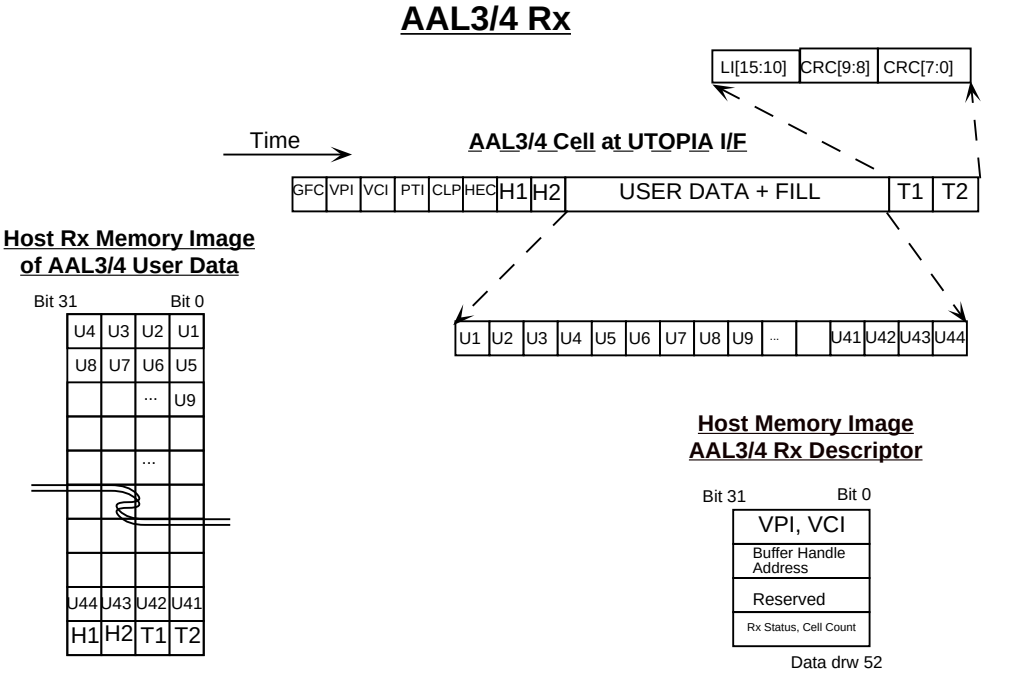
3.3.1 AAL5 Data Format in Host Memory (Little Endian)



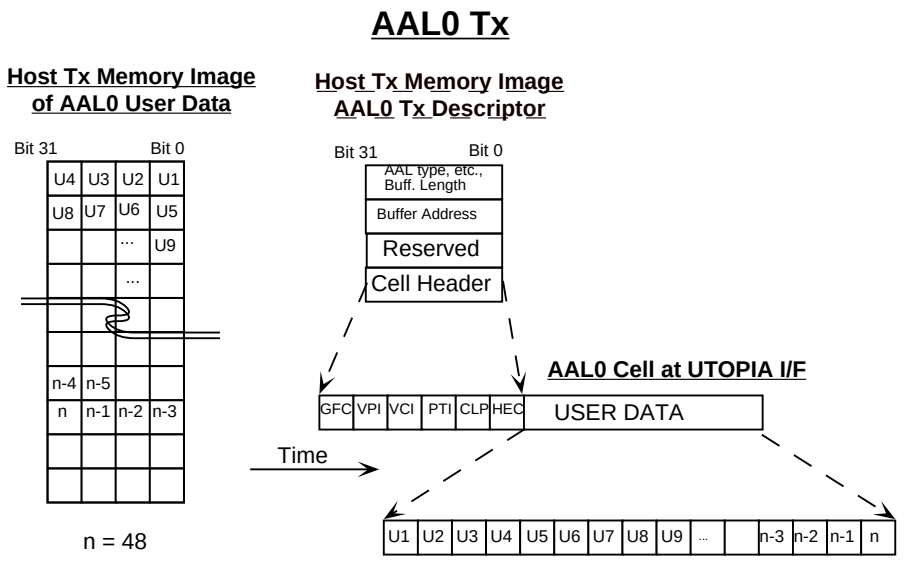


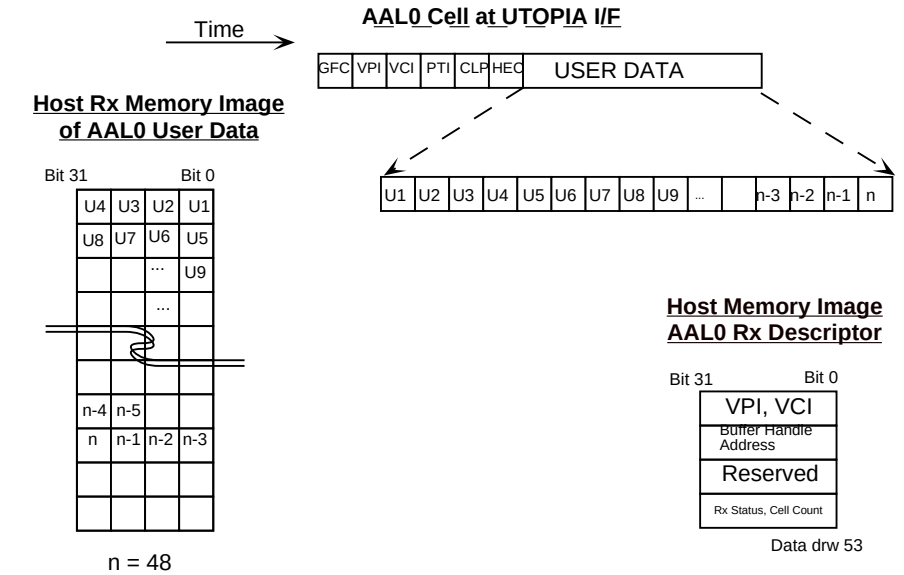
3.3.2 AAL3/4 Data Format in Host Memory





3.3.3 AAL0 Data Format in Host Memory

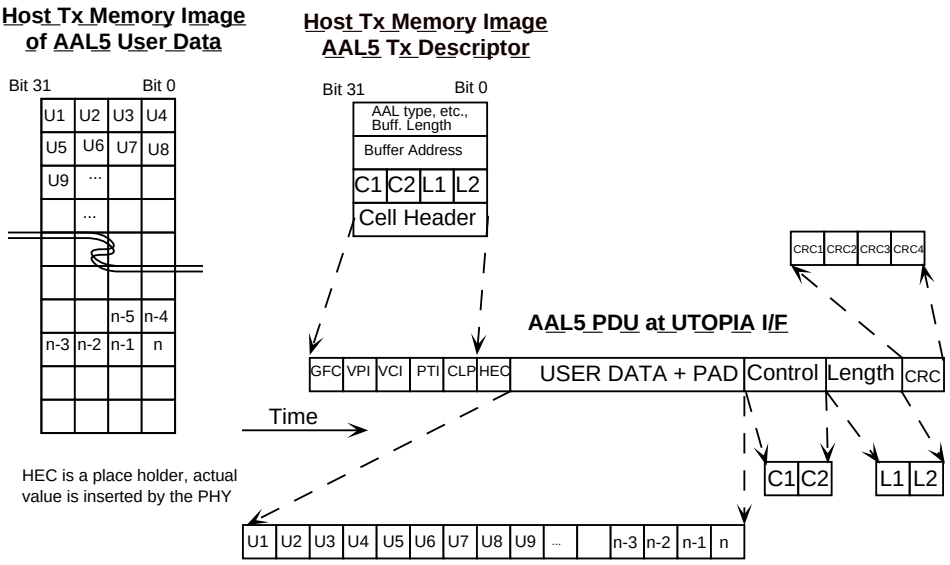




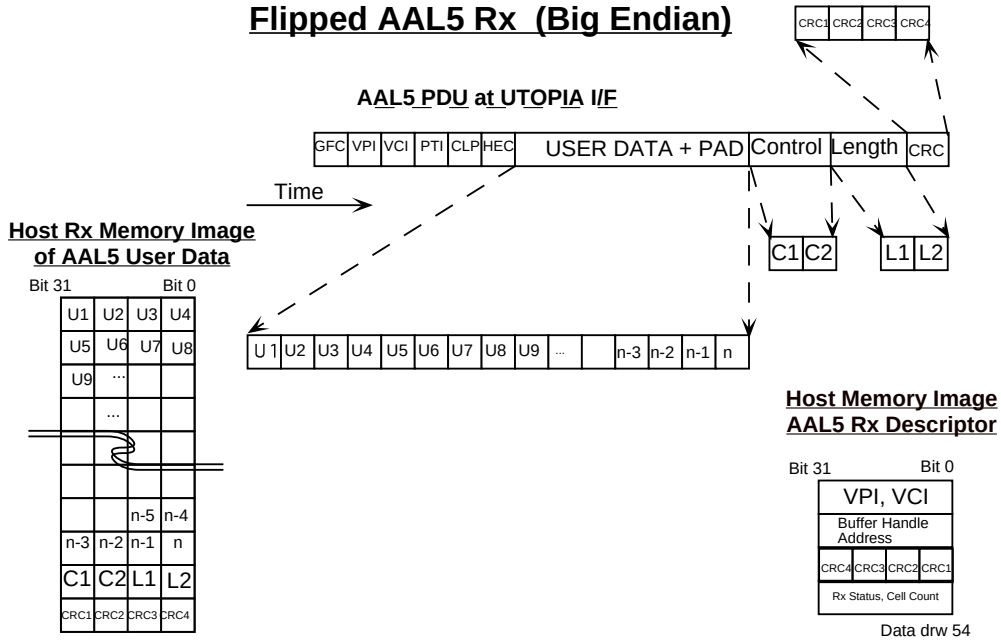
3.3.4 AAL5 Data Format in Host Memory Big Endian Operation

Data can be endian swapped as it is moved across to and from host memory. The following diagrams show how various data structures and payload are transferred in the "flipped" mode. Big endian only supports dword aligned payloads Please see the General Purpose Register (register #19) for configuration detail.

Flipped AAL5 Tx (Big Endian)

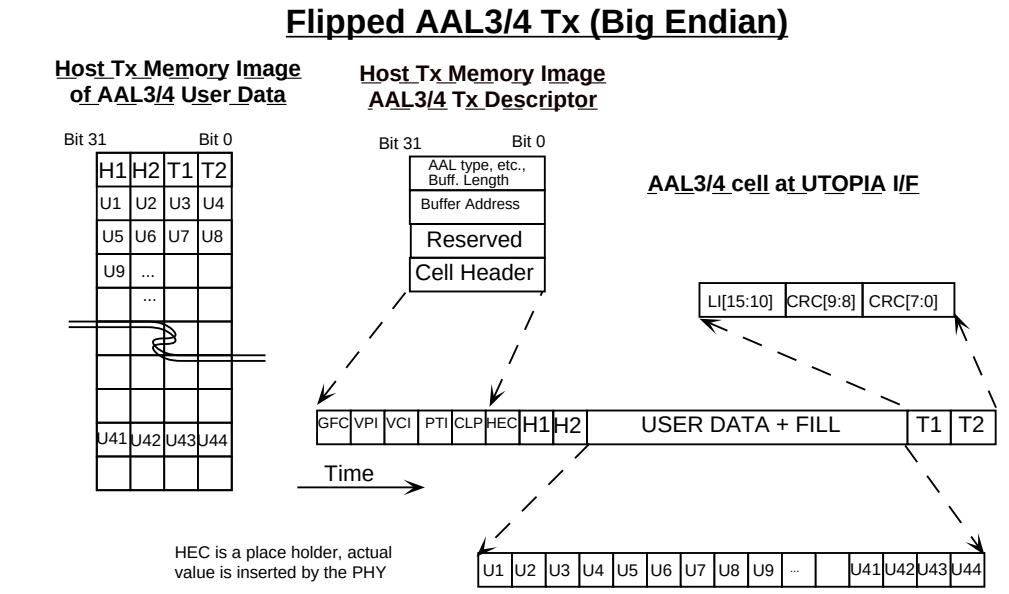


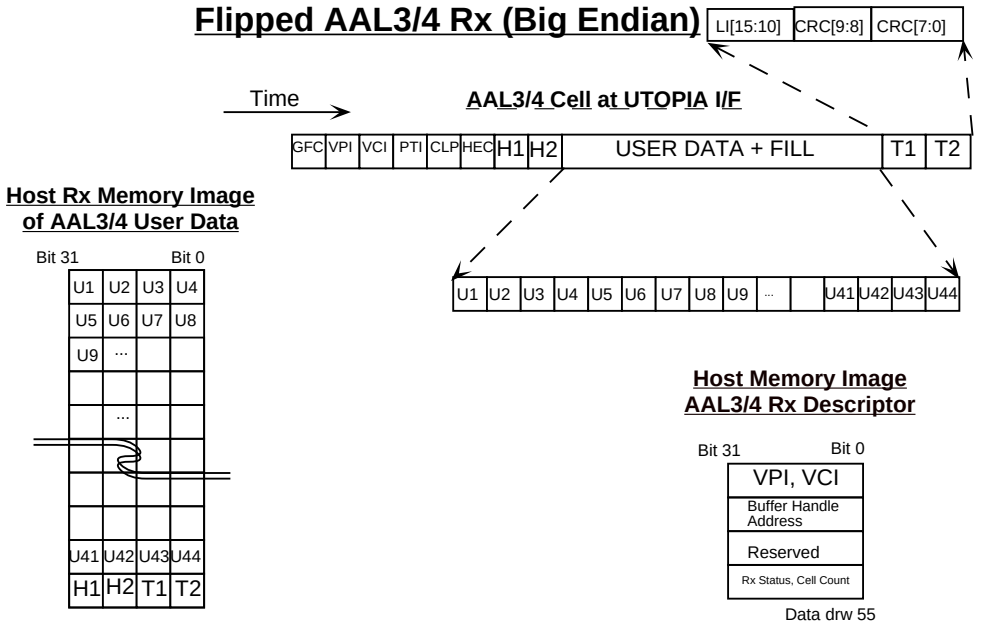
Flipped AAL5 Rx (Big Endian)



Data drw 54

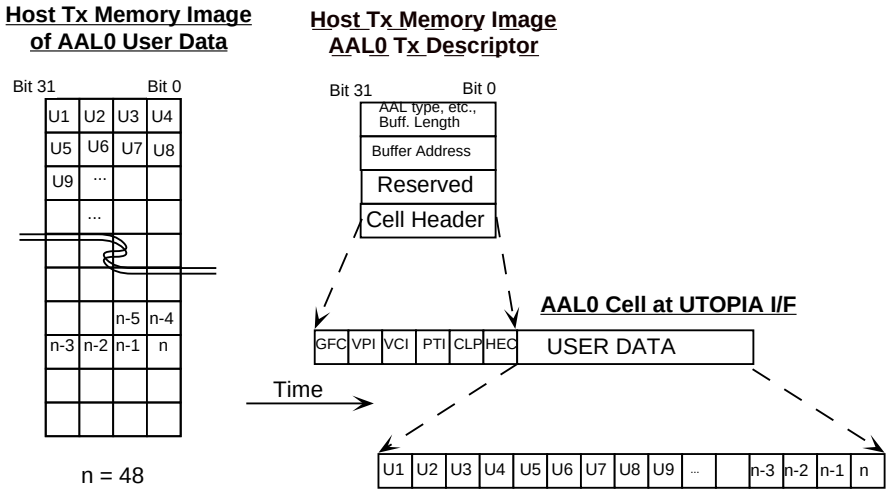
3.3.5 AAL3/4 Data Format in Host Memory Big Endian Operation



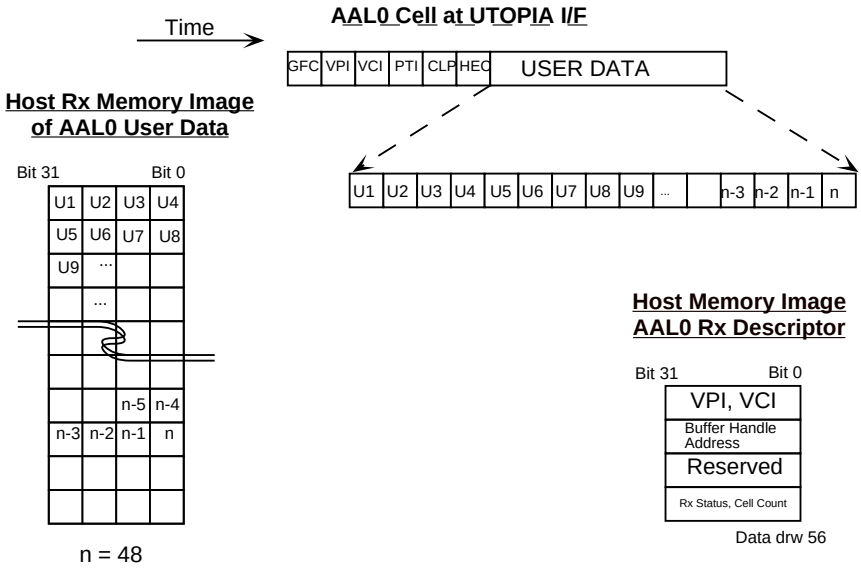


3.3.6 AAL0 Data Format in Host Memory Big Endian Operation

Flipped AAL0 Tx (Big Endian)



Flipped AAL0 Rx (Big Endian)



Data drw 56

3.4 NICStAR Network Operation Registers

This section contains the NICStAR registers that are related to the ATM network operations. The offset address is given from the memory or I/O base address as programmed in the PCI Memory or PCI I/O Base Register.

NICStAR Network Operation Register List

Reg. #	Register Name	Symbol	Offset Address (hex)	Read/ Write
0	Data register 0	DR0	0	R/W
1	Data register 1	DR1	4	W
2	Data register 2	DR2	8	W
3	Data register 3	DR3	C	W
4	Command	CMD	10	W
5	Configuration	CFG	14	R/W
6	Status	STAT	18	R/W
7	Receive Status Queue Base	RSQB	1C	W
8	Receive Status Queue Tail	RSQT	20	R
9	Receive Status Queue Head	RSQH	24	W
10	Cell Drop Counter	CDC	28	R/clear
11	VPI/VCI Lookup Error Count	VPEC	2C	R/clear
12	Invalid Cell Count	ICC	30	R/clear
13	Raw Cell Tail	RAWCT	34	R
14	Timer	TMR	38	R
15	TST Base	TSTB	3C	R/W
16	Transmit Status Queue Base	TSQB	40	W
17	Transmit Status Queue Tail	TSQT	44	R
18	Transmit Status Queue Head	TSQH	48	W
19	General Purpose	GP	4C	R/W
20	VPI/VCI Mask	VPM	50	W

Reg drw 01

3.4.1 Data Registers (DR0 - DR3) (Register #0–3)

The Data Registers are used by the host for writing and reading the local SRAM through the NICStAR. SRAM read operation can only be performed on one word at a time, while write operation can perform up to 4 words at a time.

REGISTER #0: Data Register 0 (DR0)

OFFSET	BIT FIELDS
0(H)	31-0
	data

REGISTER #1: Data Register 1 (DR1)

OFFSET	BIT FIELDS
4(H)	31-0
	data

REGISTER #2: Data Register 2 (DR2)

OFFSET	BIT FIELDS
8(H)	31-0
	data

REGISTER #3: Data Register 3 (DR3)

OFFSET	BIT FIELDS
C(H)	31-0
	data

BIT FIELD	FUNCTION	DESCRIPTION
31-0	data	These registers are used by the host to exchange parameters with the NICStAR hardware. When passing parameters to the NICStAR, the host must first load the data registers followed by issuing a command to the Command Register. When reading parameters from the NICStAR, the host must first issue a read command to the Command Register followed by polling the Command Busy bit in the Status Register. The host can read the data out from the Data Register(s) after the Command Busy bit becomes zero.

Reg drw 02

For writing data to the SRAM, the host first loads up the data into the Data Registers with 1 to 4 words, then issues a Write_SRAM command to the Command Register. For reading data from the local SRAM, the host first loads the reading address of the SRAM and a Read_SRAM command in the Command Register, then polls the Command Busy till it becomes zero, by that time the host can read the data from Data Register 0.

3.4.2 Command Register (CMD) (Register #4)

The Command Register is used to issue commands to the NICStAR. It consists of an Op-code field and a parameter field. Some commands require additional parameters in one or more Data registers. Before any command is issued, the device driver must read the Command Busy bit in the Status Register to become zero. This is done to make sure there is no previous command pending.

The device driver uses the NICStAR Command Register to access both the local SRAM and the utility bus interface to the PHY-TC component. For a read operation, the device driver issues the appropriate command word, and then polls the Command Busy Flag in the Status Register until it equals 0, indicating that the NICStAR has loaded its Data Register(s) with the information requested. For a write operation, the device driver first loads the NICStAR's Data Register(s) and then issues the appropriate command to transfer the information into either the local SRAM or across the utility bus to the PHY-TC component.

REGISTER #4: Command (CMD)

OFFSET	BIT FIELDS	
10(H)	31-28	27-0
	Opcode.	parameters

BIT FIELD	FUNCTION	DESCRIPTION
31-28	Opcode.	Command Opcode.

31	30	29	28	Hex	Command
0	0	0	0	0	No operation
0	0	0	1	1	Reserved
0	0	1	0	2	open/close_connection
0	0	1	1	3	Reserved
0	1	0	0	4	Write_SRAM
0	1	0	1	5	Read_SRAM
0	1	1	0	6	Write_FreeBufQ
0	1	1	1	7	Reserved
1	0	0	0	8	Read_Utility
1	0	0	1	9	Write_Utility
1	0	1	0	A	Reserved
1	0	1	1	B	Reserved
1	1	0	0	C	Reserved
1	1	0	1	D	Reserved
1	1	1	0	E	Reserved
1	1	1	1	F	Reserved

27-0	parameter	Command parameter. This command parameter is used with the command Opcode. to form a complete command to the NICStAR. Certain commands require additional parameters using the Data Register(s). The following will list all the commands with their associated parameters.

Reg drw 03

3.4.2.1 Command: NO OP (Register #4)

Register #4: **Command = NO OP**

OFFSET	BIT FIELDS	
10(H)	31-28	27-0
	0 (h)	Reserved

BIT FIELD	FUNCTION	DESCRIPTION
31-28	0 (h)	No operation.
27-0	reserved	Always set to all zero's.

Reg drw 04

3.4.2.2 Command: Open/Close_connection (Register #4)

Register #4: **Command = Open/close_connection**

OFFSET	BIT FIELDS				
10(H)	31-28	27-20	19	18-2	1-0
	2 (h)	reserved	OPEN	SRAD	RSVD

BIT FIELD	FUNCTION	DESCRIPTION
31-28	2 (h)	Open/close_connection command. This command is used to open or close a VC in the Receive Connection Table.
27-20	Reserved	Always set to all zero's.
19	OPEN	0 = Close a VC in the Receive Connection Table. 1 = Open a VC in the Receive Connection Table.
18-2	SRAD	SRAM Address. Specified the SRAM address of the VC entry in the Receive Connection Table. Range 00000 - 1FFFF (h), 128K words (32-bit) total. Since each VC entry has four (32-bit) words, each VC number must be an increment of four accordingly. For example, VC#0 is location in address 00000(h) whereas VC#3 is location in address 0000C(h), etc.
1-0	RSVD	Reserved, always set to all zero's.

Reg drw 05

3.4.2.3 Command: Write_SRAM (Register #4)

Register #4: **Command = Write_SRAM**

OFFSET	BIT FIELDS			
	31-28	27-19	18-2	1-0
10(H)	4 (h)	Reserved	SRAD	BSIZE

BIT FIELD	FUNCTION	DESCRIPTION
31-28	4 (h)	Write to SRAM command. The device driver uses this command to write one to four words into the local SRAM, such as SCD for Tx operation. The device driver first loads the Data Register(s) with information, and then issues this command to transfer the information into the local SRAM.
27-19	reserved	Always set to all zero's.
18-2	SRAD	SRAM Address. Specified the first data word address of a burst operation. If the burst size is more than one word, the NICStAR will auto-increment the SRAM address up to the burst size. Range 00000 - 1FFFF (h), 128K words (32-bit) total. Actual SRAM address space available depending on the SRAM installed, see SRAM Memory Maps for their address allocations.
1-0	BSIZE	Burst Size. Specifies the number of data words the NICStAR will transfer from the Data Register(s) to local SRAM. bit1 bit0 Operation 0 0 1 word write from Data Register 0 0 1 2 word write from Data Register 0 and 1 1 0 3 word write from Data Register 0, 1 and 2 1 1 4 word write from Data Register 0, 1,2 and 3

Reg drw 06

3.4.2.4 Command: Read_SRAM (Register #4)

Register #4: **Command = Read_SRAM**

OFFSET	BIT FIELDS			
	31-28	27-19	18-2	1-0
10(H)	5 (h)	Reserved	SRAD	RSVD

BIT FIELD	FUNCTION	DESCRIPTION
31-28	5 (h)	Read_SRAM command. The device driver uses this command to read one 32-word from the local SRAM, such as an entry for the Transmit Schedule Table. The device driver issues this command, and then "polls" the Command Busy bit in the Status Register until it becomes 0, at which time it may access the desired information in Data Register 0 (DR0).
27-19	reserved	Always set to all zero's.
18-2	SRAD	SRAM Address. Specified the word address of the local SRAM for this read operation. Range 00000 - 1FFFF (h), 128K words (32-bit) total. Actual SRAM address space available depending on the SRAM installed, see SRAM Memory Maps for their address allocations.
1-0	RSVD	Always set to all zero's.

Reg drw 07

3.4.2.5 Command: Write_FreeBufQ (Register #4)

Register #4: **Command = Write_FreeBufQ**

OFFSET	BIT FIELDS		
10(H)	31-28	27-1	0
	6 (h)	Reserved	LBUF

BIT FIELD	FUNCTION	DESCRIPTION
31-28	6 (h)	Write_FreeBufQ command. To add Small or Large Free Buffers to the Free Buffers queues in local SRAM, the device driver MUST use this command. The device driver first loads the four NICStAR Data Registers with either two Small FBDs or two Large FBDs (each FBD includes a Buffer Handle and a DMA Address), and then issues this command, which causes the NICStAR to transfer the contents of it's Data registers into the appropriate buffer queue. When using this command, the format for the contents of Data Registers is as follows: Data Reg 0: Buffer Handle of Free Buffer n Data Reg 1: DMA Address of Free Buffer n Data Reg 2: Buffer Handle of Free Buffer n+1 Data Reg 3: DMA Address of Free Buffer n+1 Two Free Buffers must be written at a time (all 4 Registers).
27-1	Reserved	Always set to all zero's.
0	LBUF	Large Buffer indicator. Specified the type of the FBD's in this operation. 0 = Small Free Buffer Queue. 1 = Large Free Buffer Queue.

Reg drw 08

3.4.2.6 Command: Read_Utility (Register #4)

Register #4: **Command = Read_Utility**

OFFSET	BIT FIELDS				
	31-28	27-10	9	8	7-0
10(H)	8 (h)	Reserved	UTCS1	UTCS0	UTLADD

BIT FIELD	FUNCTION	DESCRIPTION
31-28	8 (h)	Read_Utility command. This command allows the device driver to read one byte of data from the Utility bus, which interfaces to the PHY-TC component. The device driver issues this command, and then "polls" the Command Busy bit in the Status Register until it equals 0, at which time it may access the desired information in Data Register 0 bits 7-0.
27-10	reserved	Always set to all zero's.
9	UTCS1	UTL_CS1 signal. 0 = Utility bus UTL_CS1# is NOT selected. 1 = Utility bus UTL_CS1# is selected, and it will remain active for the duration of read operation.
8	UTCS0	UTL_CS0 signal. 0 = Utility bus UTL_CS0# is NOT selected. 1 = Utility bus UTL_CS0# is selected, and it will remain active for the duration of read operation.
7-0	UTLADD	Utility bus address. Specifies the byte address for the information on the utility bus.

Reg drw 09

3.4.2.7 Command: Write_Utility (Register #4)

Register #4: **Command = Write_Utility**

OFFSET	BIT FIELDS				
	31-28	27-10	9	8	7-0
10(H)	9 (h)	Reserved	UTCS1	UTCS0	UTLADD

BIT FIELD	FUNCTION	DESCRIPTION
31-28	9 (h)	Write_Utility command. This command allows the device driver to write one byte of data to the Utility bus, which interfaces to the PHY-TC component. The device driver first load the Data Register 0 (bits 7-0) with the data byte to be written, and the issues this command which will cause the data transferred to the utility bus.
27-10	Reserved	Always set to all zero's.
9	UTCS1	UTL_CS1 signal. 0 = Utility bus UTL_CS1# is NOT selected. 1 = Utility bus UTL_CS1# is selected, and it will remain active for the duration of write operation.
8	UTCS0	UTL_CS0 signal. 0 = Utility bus UTL_CS0# is NOT selected. 1 = Utility bus UTL_CS0# is selected, and it will remain active for the duration of write operation.
7-0	UTLADD	Utility bus address. Specifies the byte address for the information on the utility bus.

Reg drw 10

3.4.3 Configuration Register (CFG) (Register #5)

REGISTER #5: CONFIGURATION (CFG)

OFFSET	BIT FIELDS										
14(H)	31	30	29	28-27	26-25	24	23-22	21	20	19-18	17-16
	SWRST	RSVD	RXPTH	SMBUF	LGBUF	EFBIE	RXSTQ	ICAPT	IGGFC	VPVCS	RXCNS

BIT FIELDS													
15	14-12	11	10	9	8	7	6	5	4	3	2	1	0
VPECA	RXINT	RAWIE	RQFIE	RXRM	RSVD	TMOIE	RSVD	TXEN	TXINT	TXUIE	UMODE	TXSFI	PHYIE

BIT FIELD	FUNCTION	DESCRIPTION
31	SWRST	S/W Reset. Default = 0. 0 = No reset. 1 = Reset NICStAR. All internal registers except PCI Configuration Registers will be reloaded with the default values after this reset. When set, this function is equivalent to the PCI bus H/W reset. Device driver must wait at least two (2) PCI clocks before clearing this bit to effect a reset. The chip must be re-initialized after S/W reset before returning to its normal operation . After H/W reset, this bit is returned to 0 by the NICStAR H/W when the reset process is complete.
30	RSVD	Reserved. Must be zero.
29	RXPTH	Receive Path Enable. Default = 0. 0 = Disable receive path. No Rx cells are accepted from the PHY by the NICStAR. 1 = Enable receive path. Allow Rx cells to enter Rx cell FIFO. If disabling during a receive cell operation, the NICStAR will completely receive the cell and then stop the the receive operation.

Reg drw 11

REGISTER #5: CONFIGURATION (CFG), Offset = 14(h) - continued

BIT FIELD	FUNCTION	DESCRIPTION
28-27	SMBUF	Small Receive Buffer Size. Specifies the size of each buffer in the Small Buffer Pool. 00 = 48 bytes. Default. 01 = 96 bytes 10 = 240 bytes 11 = 2K Bytes.
26-25	LGBUF	Large Receive Buffer Size Specifies the size of each buffer in the Large Buffer Pool. 00 = 2048 bytes. Default. 01 = 4096 bytes 10 = 8192 bytes 11 = 16384 bytes.
24	EFBIE	Empty Free Buffer Queue Interrupt Enable. Default = 0. 0 = No interrupt to the host when Small and/or Large Free Buffer Queue is empty. 1 = Generate interrupt to the host when Small and/or Large Free Buffer Queue is empty.
23-22	RXSTQ	Receive Status Queue Size. Specifies the size of the Receive Status Queue. 00 = 2048 bytes. Default. 01 = 4096 bytes 10 = 8192 bytes 11 = Reserved.
21	ICAPT	Invalid Cell Accept. Default = 0. 0 = Invalid cells (GFC is Non-zero in cell header) will be discarded and the Invalid Cell Counter will be incremented. 1 = Invalid cells will be received into the Raw Cell Queue and the Invalid Cell Counter will not be incremented.
20	IGGFC	Ignore GFC. Default = 0. 0 = GFC field in the receiving cells are checked according to the ICAPT (bit 21) setting above. 1 = GFC field in the receiving cells are ignored.

Reg drw 12

REGISTER #5: CONFIGURATION (CFG), Offset = 14(h) - continued

BIT FIELD	FUNCTION	DESCRIPTION
19-18	VPVCS	VPI/VCI Select. Default = 00. This field is used in conjunction with the Receive Connection Table Size (below) to select the VPI and VCI bit range to index the Receive Connection Table. VPVCS / Rx Conn. Size / VPI bits / VCI bits 00 / 4K / none / 11:0 8K / none / 12:0 16K / none / 13:0 01 / 4K / 0 / 10:0 8K / 0 / 11:0 16K / 0 / 12:0 10 / 4K / 1:0 / 9:0 8K / 1:0 / 10:0 16K / 1:0 / 11:0 11 / 4K / 7:0 / 3:0 8K / 7:0 / 4:0 16K / 7:0 / 5:0 The VPI field in the receiving cell header is 8-bit (7:0), and the VCI field is 16-bit (15:0). The remaining portion of the VPI and VCI is checked against the VPI/VCI Mask Register to make a complete VPI/VCI comparison on the receiving cells.
17-16	RXCNS	Receive Connection Table Size. Default = 00. 00 = 4096 (4K) entries. 01 = 8192 (8K) entries. 10 = 16384 (16K) entries. 11 = Reserved.
15	VPECA	VPI/VCI Error Cell Accept. Default = 0. 0 = Discard cells that either VPI/VCI do not map into the entries in the Receive Connection Table or VPI/VCI that don't have "open" connection. 1 = Accept cells into Raw Cell Queue either VPI/VCI do not map into the entries in the Receive Connection Table or VPI/VCI that don't have "open" connection.

Reg drw 13

REGISTER #5: CONFIGURATION (CFG), Offset = 14(h) - continued

BIT FIELD	FUNCTION	DESCRIPTION
14-12	RXINT	End of Receive PDU Interrupt Handling. Default = 000. This field specifies how interrupt is generated to the host at the end of receiving a CS_PDU for AAL0, AAL3/4 AND AAL5. 000 = No interrupt is generated. 001 = Generate interrupt after 0us. 010 = Generate interrupt after 314us. 011 = Generate interrupt after 624us. 100 = Generate interrupt after 899us. 101 = Reserved. 110 = Reserved. 111 = Reserved.
11	RAWIE	Raw Cell Queue Interrupt Enable. Default = 0. This bit field is used as a global enable for generating interrupt to the host for all the VC's. In addition to this global enable bit, each VC has a separate Raw Cell Interrupt Enable bit in the Receive Connection Table. Raw cell interrupt is generated on a per VC basis. This bit and the Raw Cell Interrupt Enable bit in the Receive Connection Table must be set in order to generate interrupt to the host. 0 = No interrupt generated to the host when raw cell is received. 1 = Generate interrupt to the host when a raw cell is received into the Raw Cell Queue and the Raw Cell Interrupt Enable bit in the Receive Connection Table for the VC is set.
10	RQFIE	Receive Queue Almost Full Interrupt Enable. Default = 0. 0 = No interrupt is generated when the Receive Status is 7/8 full. 1 = Generate an interrupt when the Receive Status is 7/8 full.
9	RXRM	Receive RM Cells. Default = 0. 0 = Discard cell when cell header PTI field = 110 or 111. 1 = Put cells in Raw cell Queue when cell header PTI field = 110 or 111.

Reg drw 14

REGISTER #5: CONFIGURATION (CFG), Offset = 14(h) - continued

BIT FIELD	FUNCTION	DESCRIPTION
8	RSVD	Reserved. Must be zero.
7	TMOIE	Timer Roll Over Interrupt Enable. Default = 0. 0 = No interrupt generated when the Timer Register rolls over. 1 = Generate an interrupt to the host when the Timer Register rolls over.
6	RSVD	Reserved. Must be zero.
5	TXEN	Transmit Operation Enable. Default = 0. 0 = Transmit section is disabled, however, the state of the transmit section is preserved. 1 = The transmit section of the NICStAR is enable. This is the normal operation. If the NICStAR is enabled again from being disabled, the transmit section will resume operation from where it was disabled last time.
4	TXINT	Transmit status Interrupt Enable. Default = 0. 0 = No interrupt is generated after the NICStAR writes a Transmit Status Indicator into the Transmit Status Queue. 1 = Generate an interrupt after the NICStAR writes a Transmit Status Indicator into the Transmit Status Queue, only if the interrupt bit in the TSR is also set.

Reg drw 15

REGISTER #5: CONFIGURATION (CFG), Offset = 14(h) - continued

BIT FIELD	FUNCTION	DESCRIPTION
3	TXUIE	Transmit Under-run Interrupt Enable. Default = 0. 0 = **No** interrupt when the NICStAR starts a CS_PDU and runs out of transmit buffers before End_of_PDU occurs. 1 = Generates an interrupt when the NICStAR starts a CS_PDU and runs out of transmit buffers before End_of_PDU occurs.
2	UMODE	UTOPIA Mode Select. Default = 0. 0 = UTOPIA interface is in cell mode. 1 = UTOPIA interface is in byte mode.
1	TXSFI	Transmit Status Full Interrupt Enable. Default = 0. 0 = **No** interrupt when the Transmit Status Queue is 7/8 full. 1 = Generates an interrupt when the Transmit Status Queue is 7/8 full.
0	PHYIE	PHY Interrupt Enable. Default = 0. 0 = PHY_INT signal does **not** interrupt the host. 1 = PHY_INT signal does interrupt the host.

Reg drw 16

3.4.4 Status Register (STAT) (Register #6)

REGISTER #6: STATUS (STAT)

OFFSET	BIT FIELDS	
	31-24	23-16
18(H)	SBFQC	LBFQC

BIT FIELDS															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSIF	TXICP	RSVD	TSQF	TMROF	PHYIF	CMDBZ	SBFQF	LBFQF	RSQF	EPDU	RAWCF	SBFQE	LBFQE	RSQF	0

BIT FIELD	FUNCTION	DESCRIPTION
31-24	SBFQC	Small Buffer Queue Count, 8-bits. This is the eight (8) most significant bits (MSB) of the Small Buffer Queue (max. =512). The LSB (bit 0) of the count is not read. Actual value = SBFQC x 2 + (one or zero).
23-16	LBFQC	Large Buffer Queue Count, 8 bits. This is the eight (8) most significant bits (MSB) of the Large Buffer Queue (max. =512). The LSB (bit 0) of the count is not read. Actual value = LBFQC x 2 + (one or zero).
15	TSIF	Transmit Status Indicator (TSI) Flag. Default = 0. 0 = No TSI has been written to the Transmit Status Queue by the NICStAR. 1 = a TSI has been written to the Transmit Status Queue by the NICStAR. Host writes a "1" to clear this bit.
14	TXICP	Transmit Incomplete PDU. Default = 0. 0 = No incomplete CS-PDU has been transmitted by the NICStAR. 1 = Incomplete CS-PDU has been transmitted by the NICStAR. If the SAR runs out of TBDs prior to an End 0f PDU counter, this bit will be set. Host writes a "1" to clear this bit.

Reg drw 17

REGISTER #6: Status (STAT), Offset = 18(h) - continued

BIT FIELD	FUNCTION	DESCRIPTION
13	RSVD	Reserved. Will be read zero.
12	TSQF	Transmit Status Queue Full. Default = 0. 0 = Transmit Status Queue is **less than** 7/8 full. 1 = Transmit Status Queue is 7/8 full. This bit is cleared by the NICStAR.
11	TMROF	Timer Overflow. Default = 0. 0 = the Timer Register is not overflow. 1 = the Timer Register has been overflow. When this bit is set to 1, the host must write a "1" to this bit in order to clear it.
10	PHYI	PHY Device Interrupt flag. Default = 0. 0 = PHY device interrupt signal **NOT** active. 1 = PHY device interrupt signal active. This bit is cleared by clearing the PHY interrupt. Then write a 1 to clear this bit.
9	CMDBZ	NICStAR Command Busy flag. Default = 0. 0 = NICStAR Data register is ready to be read/written. 1 = NICStAR Data register is **NOT** ready to be read/written. Each time the host issues a command to the NICStAR, the host must poll this flag until it is cleared before another command is isued.
8	SBFQF	Small Buffer Queue Full. Default = 0. 0 = Small Buffer Queue is **NOT** full. (at least 2 sets of Buffers) 1 = Small Buffer Queue is full. This bit is cleared by the NICStAR.
7	LBFQF	Large Buffer Queue Full. Default = 0. 0 = Large Buffer Queue is **NOT** full. 1 = Large Buffer Queue is full. This bit is cleared by the NICStAR.
6	RSQF	Receive Status Queue Full. Default = 0. 0 = Receive Status Queue is **NOT** full. 1 = Receive Status Queue is full. This bit is cleared by the NICStAR.

Reg drw 18

REGISTER #6: Status (STAT), Offset = 18(h) - continued

BIT FIELD	FUNCTION	DESCRIPTION
5	EPDU	End of PDU flag. Default = 0. 0 = a complete PDU has **NOT** been transferred to the host buffer by the NICStAR for either AAL0, AAL 3/4 or AAL 5. 1 = a complete PDU has been transferred to the host buffer by the NICStAR for either AAL0, AAL 3/4 or AAL 5. The device driver can clear this bit by writing a 1 to this bit.
4	RAWCF	Raw Cell Flag. Default = 0. 0 = a complete Raw Cell has **NOT** been transferred to the Raw Cell Queue in the host memory by the NICStAR. 1 = a complete Raw Cell has been transferred to the Raw Cell Queue in the host memory by the NICStAR. The device driver can clear this bit by writing a 1 to this bit.
3	SBFQE	Small Buffer Queue Empty. Default = 0. 0 = Small buffer queue is **NOT** empty. 1 = Small buffer queue is empty. When this bit is set, an interrupt is generated to the host if the Empty Free Buffer Queue Interrupt Enable bit in the Configuration Register is set.
2	LBFQE	Large Buffer Queue Empty. Default = 0. 0 = Large buffer queue is **NOT** empty. 1 = Large buffer queue is empty. When this bit is set, an interrupt is generated to the host if the Empty Free Buffer Queue Interrupt Enable bit in the Configuration Register is set.
1	RSQAF	Receive Status Queue Almost Full. Default = 0. 0 = Receive Status Queue is **NOT** 7/8 full. 1 = Receive Status Queue is 7/8 full. When this bit is set, an interrupt is generated to the host if the Receive Queue Almost Full Interrupt Enable bit in the Configuration Register is set.
0	-----	Reserved, always reads 0.

Reg drw 19

3.4.5 Receive Status Queue Base (RSQB) (Register #7)

REGISTER #7: Receive Status Queue Base (RSQB)

OFFSET	BIT FIELDS			
1C(H)	31-13	12	11	10-0
	RSQBA			Reserved

BIT FIELD	FUNCTION	DESCRIPTION
31-13, 12 or 11	RSQBA	Receive Status Queue (RSQ) Base Address. This field specifies the starting address for the Receive Status Queue in the host memory. This field is loaded with a default value after reset. Device driver can load a different value during device initialization after reset and may not change it during run-time. Bit usage is defined as follows: Size of RSQ / Base Addr / Reserved bit field 2048 bytes (128 entries) 31-11 10-0 4096 bytes (256 entries) 31-12 11-0 8192 bytes (512 entries) 31-13 12-0 Each entry in the RSQ is four (4) 32-bit words, or 16 bytes. Defined in bits 22 and 23 of the Configuration Register (Register #5) Reserved bit field must be set to all zero.
12,11 or 10-0	Reserved	Always set to all zero's.

Reg drw 20

3.4.6 Receive Status Queue Tail (RSQT)

REGISTER #8: Receive Status Queue Tail (RSQT)

OFFSET	BIT FIELDS				
20(H)	31-13	12	11	10-2	1-0
	RSQB			RSQTA	RSVD

BIT FIELD	FUNCTION	DESCRIPTION
31-13, 12 or 11	RSQB	Receive Status Queue (RSQ) Base. This field specifies the starting address for the Receive Status Queue in the host memory. This field is the same value as the value in the Receive Status Queue Base register. It is repeated here to facilitate the host to have one read operation in order to get the complete Receive Status Tail pointer. Bit usage depending on the size of RSQ, see table below.
12,11 or 10-2	RSQTA	Receive Status Queue Tail pointer offset. This field is read together with RSQB field above to form a complete physical address (32-bit word boundary) for the Receive Status Queue Tail pointer to the host memory. Bit usage is defined as follows: Size of RSQ — RSQB field — RSQTA field 2048 bytes — 31-11 — 10-2 4096 bytes — 31-12 — 11-2 8192 bytes — 31-13 — 12-2 RSQ size is set in the Configuration Register (Register #5) bits 22 and 23. The NICStAR updates this Tail pointer offset to reflect the last entry valid in the Receive Status Queue.
1-0	RSVD	Reserved. Always set to zero.

Reg drw 21

3.4.7 Receive Status Queue Head (RSQH) (Register #9)

REGISTER #9: Receive Status Queue Head (RSQH)

OFFSET	BIT FIELDS				
	31-13	12	11	10-2	1-0
24(H)	Reserved			RSQHA	RSVD

BIT FIELD	FUNCTION	DESCRIPTION
31-13, 12 or 11	Reserved	Reserved. Always be zero. The actual number of reserved bits depend on the size of the Receive Status Queue (RSQ), see below for the bit usage for this field.
12, 11 or 10 - 2	RSQHA	Receive Status Queue Head pointer offset. This field is written by the device driver to indicate the last entry in the RSQ that the device driver has serviced. At initialization, this register should br cleared. This field is used together with the Receive Status Queue Base to form a complete physical address (32-bit word boundary) for the Receive Status Queue Head pointer to the host memory. Bit usage is defined as follows: Size of RSQ — Reserved — RSQHA field 2048 bytes — 31-11 — 10-2 4096 bytes — 31-12 — 11-2 8192 bytes — 31-13 — 12-2 RSQ size is set in Configuration Register (Register #5) bits 22 and 23. All reserved bits must be zero's.
1-0	RSVD	Reserved. Always set to zero.

Reg drw 22

3.4.8 Cell Drop Count (CDC) (Register #10)

REGISTER #10: Cell Drop Count (CDC)

OFFSET	BIT FIELDS	
28(H)	31-16	15-0
	Reserved	CDCNT

BIT FIELD	FUNCTION	DESCRIPTION
31-16	Reserved	Reserved. Always be zero.
15-0	CDCNT	Cell Drop Count. Default = 0000(h). The NICStAR increments this register whenever the 315-cell Rx FIFO is full, and thus is unable to accept a new received cell. This register will contain a non-zero value when the PCI bus utilization of other PCI bus masters is excessively high, thus delaying the NICStAR from transferring received cell payloads in the 315-cell Rx FIFO to host memory. The value of this register will be cleared to zero each time, after being read by the device driver.

Reg drw 23

3.4.9 VPI/VCI Lookup Error Count (VPEC) (Register #11)

REGISTER #11: VPI/VCI Lookup Error Count (VPEC)

OFFSET	BIT FIELDS	
	31-16	15-0
2C(H)	Reserved	VPLUEC

BIT FIELD	FUNCTION	DESCRIPTION
31-16	Reserved	Reserved. Always be zero.
15-0	VPLUEC	VPI/VCI Look-up Error Count. Default = 0000(h). The NICStAR increments this register if either of the following two conditions occur: 1.) the received cell's VPI/VCI field does not map into the Receive Connection Table, or 2.) the received cell's VPI/VCI field does map into the Receive Connection Table, but the Open/Close bit field for the VC is set to Close. If the VPI/VCI Error Cell Accept bit field is set to 1 (in Configuration Register), each cell received will be transferred directly into the Raw Cell Queue in host memory. If the VPI/VCI Error Cell Accept bit field is equal to 0 (default), the counter will be incremented and each cell received with either of the above conditions, will be discarded (they will not be stored into the Raw Cell Queue). The value of this register will be cleared to zero after each read operation by device driver.

Reg drw 24

3.4.10 Invalid Cell Count (ICC) (Register #12)

REGISTER #12: Invalid Cell Count (ICC)

OFFSET	BIT FIELDS	
	31-16	15-0
30(H)	Reserved	IVCNT

BIT FIELD	FUNCTION	DESCRIPTION
31-16	Reserved	Reserved. Always be zero.
15-0	IVCNT	Invalid Cell Count. Default = 0000(h). The NICStAR increments this register if both the GFC field of the received cell is not equal to zero and the Invalid Cell Accept bit field (in Configuration Register) is set to 1. In this case, the cell received will be transferred directly into the Raw Cell Queue in host memory. If the Invalid Cell Accept bit field is equal to 0 (default), this register will be incremented and the cell received will be discarded (the cell will not be stored into the Raw Cell Queue). The value of this register will be cleared to zero after each read operation by the device driver.

Reg drw 25

3.4.11 Raw Cell Tail (RAWCT) (Register #13)

REGISTER #13: Raw Cell Tail (RAWCT)

OFFSET	BIT FIELDS	
	31-6	5-0
34(H)	RAWCTA	Reserved

BIT FIELD	FUNCTION	DESCRIPTION
31-6	RAWCTA	Raw Cell Tail Address. Default = 0000(h). This register contains the value written by the NICStAR as the current tail pointer for the Raw Cell Queue in host memory (i.e., the tail pointer specifies the next available memory location in the queue in which the NICStAR will store an entry). The device driver reads this register for the Tail value and compares it with its value of the Raw Cell Head to determine if there are "unserviced" entries in the Raw Cell Queue. If the Tail value is equal to the Head value, then the Raw Cell Queue does not currently contain any "unserviced" entries, otherwise, entries need to be serviced. At initialization, this register is loaded by the NICStAR with the DMA Address value of the first Free Buffer descriptor in the local SRAM's Large Free Buffer Queue.
5-0	Reserved	Always be zero.

Reg drw 26

3.4.12 Timer (TMR) (Register #14)

REGISTER #14: Timer (TMR)

OFFSET	BIT FIELDS	
	31-24	23-0
38(H)	Reserved	TMRCNT

BIT FIELD	FUNCTION	DESCRIPTION
31-24	Reserved	Always be zero.
23-0	TMRCNT	Timer count. This register value is incremented by one every 333 SAR clocks (13.3 us when using 50MHz) by the NICStAR. When the value reaches terminal count FFFFFF(h), it will roll over to 000000(h), at which point the NICStAR writes a Timer Roll Over descriptor into the Transmit Status Queue. The timer rolls over approx. once every 3.72 minutes when using 50MHz clock. When the Timer Register rolls over, the NICStAR will generate an interrupt to the host if the Timer Roll Over Interrupt Enable is set to 1 (bit 7 in Configuration Register).

Reg drw 27

3.4.13 TST Base (TSTB) (Register #15)

REGISTER #15: Transmit Schedule Table Base (TSTB)

OFFSET	BIT FIELDS		
	31-19	18-2	1-0
3C(H)	RSVD	TSTBA	RSVD

BIT FIELD	FUNCTION	DESCRIPTION
31-19	RSVD	Reserved. Always be zero.
18-2	TSTBA	Transmit Schedule Table Base address. Default = 0000(h). This register is written by the device driver to specify the base or start address of the Transmit Schedule Table (TST) in local SRAM. This value should not be changed after the Tx operation is enabled.
1-0	RSVD	Reserved. Always be zero.

Reg drw 28

3.4.14 Transmit Status Queue Base (TSQB) (Register #16)

REGISTER #16: Transmit Status Queue Base (TSQB)

OFFSET	BIT FIELDS	
	31-13	12-0
40(H)	TSQBA	Reserved

BIT FIELD	FUNCTION	DESCRIPTION
31-13	TSQBA	Transmit Status Queue (TSQ) Base Address. This field specifies the starting address for the Transmit Status Queue in the host memory. This field is loaded with a default value after **reset**. Device driver can load a different value during device initialization after reset and may not change it during run-time. This field must be a multiple of 8192 bytes. Default value = 00000 (h).
12-0	Reserved	Always set to all zero's.

Reg drw 29

3.4.15 Transmit Status Queue Tail (TSQT) (Register #17)

REGISTER #17: Transmit Status Queue Tail (TSQT)

OFFSET	BIT FIELDS				
	31-13	12	11	10-2	1-0
44(H)	TSQB			TSQTA	RSVD

BIT FIELD	FUNCTION	DESCRIPTION
31-13, 12 or 11	TSQB	Transmit Status Queue (TSQ) Base. This field specifies the starting address for the Transmit Status Queue in the host memory. This field is the same value as the value in the Transmit Status Queue Base register. It is repeated here to facilitate the host to have one read operation in order to get the complete Transmit Status Tail pointer.
12, 11, or 10-2	TSQTA	Transmit Status Queue Tail address. This field is read together with TSQB field above to form a complete physical address (32-bit word boundary) for the Transmit Status Queue Tail pointer to the host memory. The Tail will always point to address XXXXXXXO. The NICStAR updates this Tail pointer offset to reflect the last entry valid in the Transmit Status Queue.
1-0	RSVD	Reserved. Always set to zero.

Reg drw 30

3.4.16 Transmit Status Queue Head (TSQH) (Register #18)

REGISTER #18: Transmit Status Queue Head (TSQH)

OFFSET	BIT FIELDS		
48(H)	31-13	12-2	1-0
	Reserved	TSQHA	RSVD

BIT FIELD	FUNCTION	DESCRIPTION
31-13	Reserved	Reserved. Always be zero.
12-2	TSQHA	Transmit Status Queue Head address offset. This field is written by the device driver to indicated the last entry in the TSQ that the device driver has serviced. At initialization, this register should be reset. This field is used together with the Transmit Status Queue Base to form a complete physical address (32-bit word boundary) for the Transmit Status Queue Head pointer to the host memory.
1-0	RSVD	Reserved. Always set to zero.

Reg drw 31

3.4.17 General Purpose (GP) (Register #19)

The General Purpose Register is used for three different types of operations:

1) to clock information out of or into the external serial EEPROM,

2) to reset the PHY-TC component, and

3) accumulating the Tx negative credit count.

When clocking information out of or into the serial EEPROM, the device driver controls the waveform of the EEPROM's clock input and the polarity level of the EEPROM's chip select input via bit fields in the General Purpose Register. When reading information from the EEPROM, the device driver reads the EEPROM Data In bit field (bit 16) of the General Purpose Register; the value in this bit field corresponds to the polarity level of the serial EEPROM's output pin. When writing information to the EEPROM, the device driver writes to the EEPROM Data Out bit field (bit 0) of the General Purpose Register; the value in this bit field determines the polarity of the serial EEPROM's input pin.

REGISTER #19: General Purpose (GP)

OFFSET	BIT FIELDS								
4C(H)	31-24	23-17	16	15	14-4	3	2	1	0
	TXNCC	RSVD	EEDI	BIGE	RSVD	PHYRST	EESCLK	EECS	EEDO

BIT FIELD	FUNCTION	DESCRIPTION
31-24	TXNCC	Tx Negative Credit Count. This register specifies the number of cell times the NICStAR's 9-cell Tx Cell-Out FIFO is empty). It will count up to OXFF and wrap around.
23-17	RSVD	Always be zero.
16	EEDI	EEPROM Data In. The value in this bit field corresponds to the polarity level of the NICStAR's input pin which should be connected to the serial EEPROM's output pin.
15	BIGE	Big Endian Operation 0 = PCI Data transfered in Little Endian. 1 = PCI data transfered in Big Endian.
14 - 4	Reserved	Always be zero.
3	PHYRST	PHY Reset. If equal to 0 (default), the PHY_RST# pin is set to high. If set to 1, the PHY_RST# pin becomes asserted (i.e., it becomes equal to 0).

Reg drw 32

REGISTER #19: General Purpose (GP), Offset = 4C(h) - continued

BIT FIELD	FUNCTION	DESCRIPTION
2	EESCLK	EEPROM Clock. The value in this bit field corresponds to the polarity level of the serial EEPROM's clock input. To clock information out of or into the serial EEPROM, the device driver will need to send a 0-1-0-1- etc. transition to this bit field, while simultaneously performing the EEPROM access.
1	EECS	EEPROM Chip Select. The value in this bit field determines the polarity of the serial EEPROM's chip select input pin.
0	EEDO	EEPROM Data Out. The value in this bit field determines the polarity of the NICStAR's output pin which should be connected to the serial EEPROM's data input pin.

Reg drw 33

3.4.18 VPI/VCI Mask (VPM) (Register #20)

REGISTER #20: VPI/VCI Mask (VPM)

OFFSET	BIT FIELDS	
	31-12	11-0
50(H)	Reserved	VPCMK

BIT FIELD	FUNCTION	DESCRIPTION
31-12	Reserved	Always be zero.
11-0	VPCMK	VPI/VCI MSb Mask. The device driver writes the MSB portions of both the VPI and VCI fields for all VCs into this register. These portions become a mask which compare with the corresponding MSB portions of a received cell's VPI and VCI fields. The Receive Connection Table contains the LSB portions of both the VPI and VCI fields for all open VCs. Because OAM cells are received along with VPI/VCI value, this register should normally be set to zero. Setting a non-zero value in this field may not receive OAM cells defined by the ATM Forum.

Reg drw 34

3.5 PCI Configuration Registers

There are 256 bytes of PCI Configuration Registers. The host system BIOS software reads the Configuration Registers using the Configuration Read command when the IDSEL# and AD[7:2] signals. (AD[1:0] = 00 and AD[31:8] are ignored). These registers are all either 8, 16, or 32-bit accessible.

Table 9: PCI Configuration Register Map

Byte Offset Addr	Bit Range 31 24 23 16 15 8 7 0			
00h	Device ID = 0001h		Vendor ID = 111Dh	
04h	PCI Status		PCI Command	
08h	Device Class Code			Rev ID = 02h
0Ch	BIST	Header Type	Latency Timer	Reserved
10h	NICStAR I/O Base Address			
14h	NICStAR Memory Base Address			
18h .. 28h	Reserved			
30h	Expansion ROM Base Address			
34h 38h	Reserved			
3Ch	MAX_LAT	MIN_GNT	Interrupt Pin	Interrupt Line
40h .. FCh	Reserved			

3.5.1 PCI Configuration Register: Command/Status

PCI Configuration Register: Command/Status

OFFSET	BIT FIELDS								
	31	30	29	28	27	26-25	24	23	22-16
04(H)	PARER	0	MRABT	TGABT	RSVD	IOSPD	PARED	FSACC	RSVD

BIT FIELDS								
15 -10	9	8	7	6	5 -3	2	1	0
RSVD	FSCYC	SERRE	0	PARDE	RSVD	MSTEN	MEMEN	IOEN

BIT FIELD	FUNCTION	DESCRIPTION
31	PARER	Parity Error. 0 = No parity error was detected. 1 = At least one parity error was detected. This bit is set regardless of the setting of bit 6 in this register. This bit can be cleared by writing a "1" to this bit.
30	RSVD	Reserved. Will be read as zero.
29	MRABT	Master Abort. Default = 0. 0 = NICStAR Master bus cycles are completed successfully. 1 = DELSEL# signal from target was not activated after 7 PCI clocks. This bit can be cleared by writing a "1" to this bit.
28	TGABT	Target Abort. Default = 0. 0 = NICStAR Master bus cycles were NOT aborted by a target device. 1 = NICStAR Master bus cycles was aborted due to target aborting. This bit can be cleared by writing a "1" to this bit.
27	RSVD	Reserved Will be read as Zero.

PCI drw 02

PCI Configuration Register: Command/Status - cont.
Offset = 04(h)

BIT FIELD	FUNCTION	DESCRIPTION
26-25	IOSPD	I/O Access Decode Speed. These bits specifies how fast the NICStAR can assert DEVSEL# signal in a target operation. These bits are read only. Always read as **01** for medium decode time. i.e. one wait state.
24	PARED	Parity Error Detected. Default = 0. 0 = No parity error detected. 1 = when all of the following conditions occur: a) when NICStAR was a bus master and the PERR# was asserted by the NICStAR or the target. b) the parity error response bit (bit 6 of this reg) is set to 1.
23	FSACC	Fast Back to back Access. Always = 1. This read-only bit indicates to the host that the NICStAR as a target can accept fast back to back bus transactions when the bus transaction are not to the NICStAR.
22-16	RSVD	Reserved. Always = 00(h).

PCI drw 03

PCI Configuration Register: Command/Status - cont.
Offset = 04(h)

BIT FIELD	FUNCTION	DESCRIPTION
15-10	RSVD	Reserved. Will be read as zero.
9	FSCYS	Fast back-to-back host cycle. Default = 0. 0 = Fast back to back transfers are allowed to the same agent when NICStAR is a bus master. 1 = Fast back to back transfers are allowed to **different** agents when NICStAR is a bus master.
8	SERRE	SERR# pin enable. Default = 0. 0 = SERR# pin driver is disabled. 1 = SERR# pin driver is enabled
7	RSVD	Reserved. Will be read as zero.
6	PARDE	Parity Error Detect Enable. Default = 0. 0 = Ignore any parity error and continue normal operation. Parity error signal will still be generated. 1 = Take action when parity error is detected.
5-3	RSVD	Reserved. Will be read as zero.
2	MSTEN	NICStAR Master Enable. Default = 0. 0 = NICStAR can NOT generate master cycles. 1 = NICStAR can generate master cycles.
1	MEMEN	Memory Access to NICStAR Enable. Default = 0. 0 = NICStAR does NOT respond to PCI bus memory access. 1 = NICStAR does respond to PCI bus memory access.
0	IOEN	Memory Access to NICStAR Enable. Default = 0. 0 = NICStAR does NOT respond to PCI bus I/O access. 1 = NICStAR does respond to PCI bus I/O access.

PCI drw 04

3.5.2 PCI Configuration Register: Device Class Code

PCI Configuration Register : Class Code

OFFSET	BIT FIELDS			
	31 - 24	23 - 17	16 - 8	7 - 0
08(H)	Base Class Code	Sub-Class Code	Prog I/F	Rev ID

BIT FIELD	FUNCTION	DESCRIPTION
31 - 24	Base Class Code	Base Class Code. Will be read as 02(h). A host write to this field has no effect. The content of this register is not changed by PCI bus reset or S/W reset.
23 - 17	Sub-Class Code	Sub-Class Code. Will be read as 03(h). A host write to this field has no effect. The content of this register is not changed by PCI bus reset or S/W reset.
16 - 8	Prog I/F	Programming Interface. Will be read as 00(h). A host write to this field has no effect. The content of this register is not changed by PCI bus reset or S/W reset.
7 - 0	Rev ID	Rev ID. Will be read as 02(h). This bit field indicates the Revision level of the chip. A host write to this field has no effect. The content of this register is not changed by PCI bus reset or S/W reset.

PCI drw 05

3.5.3 PCI Configuration Register: Latency Timer

PCI Configuration Register : Latency Timer

OFFSET	BIT FIELDS			
0C(H)	31 - 24	23 - 16	15 - 8	7 - 0
	BIST	Header Type	Latency Timer	RSVD

BIT FIELD	FUNCTION	DESCRIPTION
31 - 24	BIST	Built-in Self Test. Will be read as zero, built-in self test not used.
23 - 16	Header Type	Header Type. Will be read as 00(h), to indicate the layout of the above configuration space format.
15 - 8	Latency Timer	Latency Timer. The host writes to this bit field to control the size of the master latency timer. The NICStAR's master latency timer is used in multiple of 32 PCI clocks. The master latency timer starts counting down by one PCI clock when the NICStAR is bus master and asserts FRAME# signal. If the GNT# signal is de-asserted, the NICStAR continues bus transactions. In this way the PCI bus minimum latency is guaranteed. Bits 15-13 are read/write accessible. Default = 3(h). Powerup/reset value = 0(h). The default value will be loaded after powerup or reset. Bits 12-8 are hard wired to "0 0000", read only.
7 - 0	RSVD	Reserved. Will be read as zero.

PCI drw 06

3.5.4 PCI Configuration Register: IO Base Address

PCI Configuration Register : I/O Base Address

OFFSET	BIT FIELDS		
	31 - 2	1	0
10(H)	NICStAR I/O Base Address	RSVD	1

BIT FIELD	FUNCTION	DESCRIPTION
31 - 2	NICStAR I/O Base Address	NICStAR I/O Base Address. Default = 0000(h). The host read this field to get the I/O base address. At PCI bus reset or S/W reset command, the default bit value is loaded to the register.
1	RSVD	Reserved. Will be read as 0.
0	None	Read as a 1.

PCI drw 07

3.5.5 PCI Configuration Register: Memory Base Address

PCI Configuration Register : Memory Base Address

OFFSET	BIT FIELDS	
	31 - 4	3 -0
14(H)	NICStAR Memory Base Address	RSVD

BIT FIELD	FUNCTION	DESCRIPTION
31 - 4	NICStAR Memory Base Address	NICStAR Memory Base Address. Default = 0000(h). The host writes to this register for setting the NICStAR memory base address and reads this field to get the memory base address. At PCI bus reset or S/W reset command, the default bit value is loaded to the register.
3-0	RSVD	Reserved Will be read as a zero.

PCI drw 08

3.5.6 PCI Configuration Register: Expansion ROM Base Address

PCI Configuration Register : Expansion ROM Base Address

OFFSET	BIT FIELDS		
30(H)	31 - 17	16 - 1	0
	Expansion ROM Base Address	RSVD	ADDEN

BIT FIELD	FUNCTION	DESCRIPTION
31 - 17	Expansion ROM Base Address	Expansion ROM Base Address. Default = 0000(h). The host writes to this register for setting the NICStAR expansion ROM base address and reads this field to get the expansion ROM base address. Reading of this field does not change its content. At PCI bus reset, the default bit value is loaded to the register. ROM is not accessible when the NICStAR receive is enabled.
16 - 1	RSVD	Reserved. Always read 0000(h). Writing to this field does not alter its content.
0	ADDEN	Address Decode Enable. 0 = The NICStAR does NOT allow expansion ROM access. 1 = The NICStAR does allow expansion ROM access if the Memory Space bit in PCI Command Register is also set.

PCI drw 09

3.5.7 PCI Configuration Register: Bus Grant and Interrupt

PCI Configuration Register : Bus Grant Latency and Interrupt

OFFSET	BIT FIELDS			
	31 - 24	23 - 16	15 - 8	7 - 0
3C(H)	MAX_LAT	MIN_GNT	Interrupt Pin	Interrupt Line

BIT FIELD	FUNCTION	DESCRIPTION
31 - 24	MAX_LAT	MAX_LAT. Default = 05(h). This bit field specifies how often the NICStAR needs to gain (burst) access to the PCI bus, assuming a 33MHz PCI bus clock. The period is measured in 1/4 micro second units. Reading of this bit field does not change its value and writing to this bit field has no effect. At PCI bus reset, the default value is loaded into this bit field. Worst case is when 353,773 cells are transmitted and another 353,773 cells are received simultaneously. (1/353773 * 2) * 4 = 5.6 units.
23 - 16	MIN_GNT	MIN_GNT. Default = 05(h). This bit field specifies the desired minimum period of the NICStAR on the PCI bus, assuming a 33MHz PCI bus clock. The period is measured in 1/4 micro second units. Reading of this bit field does not change its value and writing to this bit field has no effect. At PCI bus reset, the default value is loaded into this bit field. Worst case is one Tx and one Rx cells transferred back to back. At 33MHz bus clock, the burst time is = 30 * (12+16) * 4 = 4.56 (1/4 micro second units).
15 - 8	Interrupt Pin	Interrupt Pin. Default = 01(h) The bit field identifies the NICStAR is using PCI bus pin **INTA#** as its interrupt pin. Reading of this bit field does not change its value and writing to this bit field has no effect. Will be read as 01(h). At PCI bus reset, the default value is loaded into this bit field.

PCI drw 10

PCI Configuration Register : Bus Grant Latency and Interrupt - continued
Offset = 3C(h)

BIT FIELD	FUNCTION	DESCRIPTION
7 - 0	Interrupt Line	Interrupt Line. Default = 00(h) This bit field is used by the host to initialize the Interrupt Line routing information (i.e. NICStAR' s IRQ number). The host reads from this bit field to determine the NICStAR's interrupt level. Reading of this bit field does not change its value.

PCI drw 11

4 ATM Layer Rx Cell Handling

This section describes how the NICStAR will handle the received cells under different conditions.

4.1 Received Cell Accept Conditions

Table 10: Received Cell Accept Conditions

No	Rx Cell Condition	Route to Receive Buffer	Route to Raw Cell Queue	Usage
1	i) VPI/VCI **exists** in Conn. Table ii) Conn. Table Entry is **Open** iii) Conn. Table Entry is AAL0, 3/4 or 5 iv) Rx cell GFC = 0000b.	√		AAL0, 3/4, and AAL5
2	i) VPI/VCI **exists** in Conn. Table ii) Conn. Table Entry is **Open** iii) Conn. Table Entry is **Raw Cell** iv) Rx cell GFC = 0000b.		√	Raw Cell
3	i) VPI=xxh, VCI =**3 or 4**V (mask = 00h) ii) Conn. Table Entry is **Open** iii) Conn. Table Entry is **Raw Cell** iv) Rx cell GFC = 0000b.		√	**F4** OAM Cell
4	i) VPI/VCI does **NOT** exist in Connection Table ii) VPI/VCI ***Error Cell Accept*** = 1 iii) Rx cell GFC = 0000b.		√	Any cell which has VPI/ VCI value not in Connection Table
5	i) VPI/VCI **exists** in Connection Table ii) Conn. Table Entry is **Close** iii) VPI/VCI ***Error Cell Accept*** = 1 iv) Rx cell GFC = 0000b.		√	Raw Cell
6	i) Rx cell GFC !=0000b ii) Conn. Table Exists and is **OPEN** iii) Conn. Table Entry is AAL0, 3/4 or 5 iv) ***Ignore GFC*** bit =1	√		Non Raw Cell
7	i) Rx cell GFC !=0000b ii) Conn. Table Exists and is **OPEN** iii) Conn. Table Entry is **RAW Cell** iv) ***Ignore GFC*** bit =1		√	Raw Cell
8	i) VPI/VCI **exists** in Conn. Table ii) Conn. Table Entry is **Open** iii) Rx cell GFC = 0000b iv) PTI = **100b** or **101b** (OAM).		√	**F5 OAM** Cell
9	i) VPI/VCI != 000000h ii) Rx cell GFC = xxxxb iii) PTI = **110b** or **111b**. iV) CFG bit 9 = 1		√	Flow Control

Table 10: Received Cell Accept Conditions

No	Rx Cell Condition	Route to Receive Buffer	Route to Raw Cell Queue	Usage
10	i) VPI/VCI = xxxxxxh ii) Rx cell GFC **!= 0000b** iii) Invalid Cell Accept = 1 iv) Ignore GFC = 0.		√	Raw Cell

To receive a cell, the following conditions **MUST also** be met:
a) Neither NICStAR reset nor S/W reset is active
b) Rx path Enabled.
c) Rx Cell FIFO is NOT Full.
d) Free Rx buffer Available.

Cells will be dropped and not reported if there are no Free Buffers available and the Rx Cell FIFO fills up. As additional cells arrive, the first cell in the FIFO will be dropped and not reported. Insure that Free Buffers are returned in a timely fashion.

Legends used in the table:
!= means not equal to
b means binary
h means hex
x means don't care (any value)

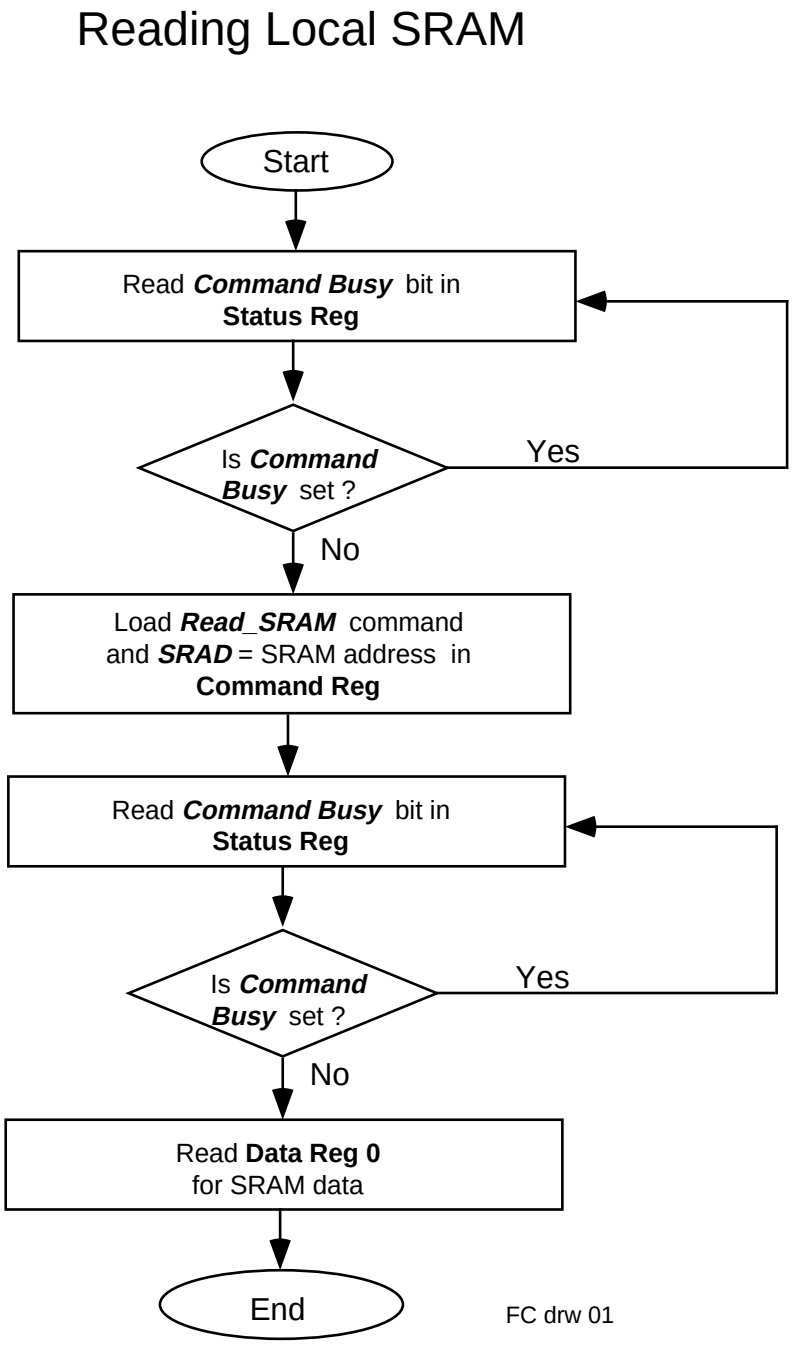
Table 11: Received Cell Discard Conditions and Error Counters

No	Rx Cell Condition	Increment Cell Drop Counter (**CDC**)	Increment VPI/VCI Lookup Error Counter (**VPEC**)	Increment Invalid Cell Counter (**ICC**)	Discard Cell
1	Rx path **Disabled**.				√
2	Rx Cell FIFO is **FULL**.	√			√
3	**No** Free Rx Buffer Available. VPI/VCI = xxxxxxh, PTI = xxxb.				√
4	i) VPI/VCI does **NOT** exist in Conn. Table ii) ***VPI/VCI Error Cell Accept*** = **0**		√		√
5	i) VPI/VCI **does** exist in Conn. Table but the Connection is **CLOSED**		√		√
6	i) Rx cell header GFC != 0000b ii) VPI/VCI=xxxxxxh iii) ***Invalid Cell Accept*** = **0** iv) ***Ignore GFC*** = 0			√	√
7	i) Rx cell header GFC = 0000b ii) VPI=00h, VCI = 0000h, PTI = xxxb iii) CLP = 0.				√ (Null cell)

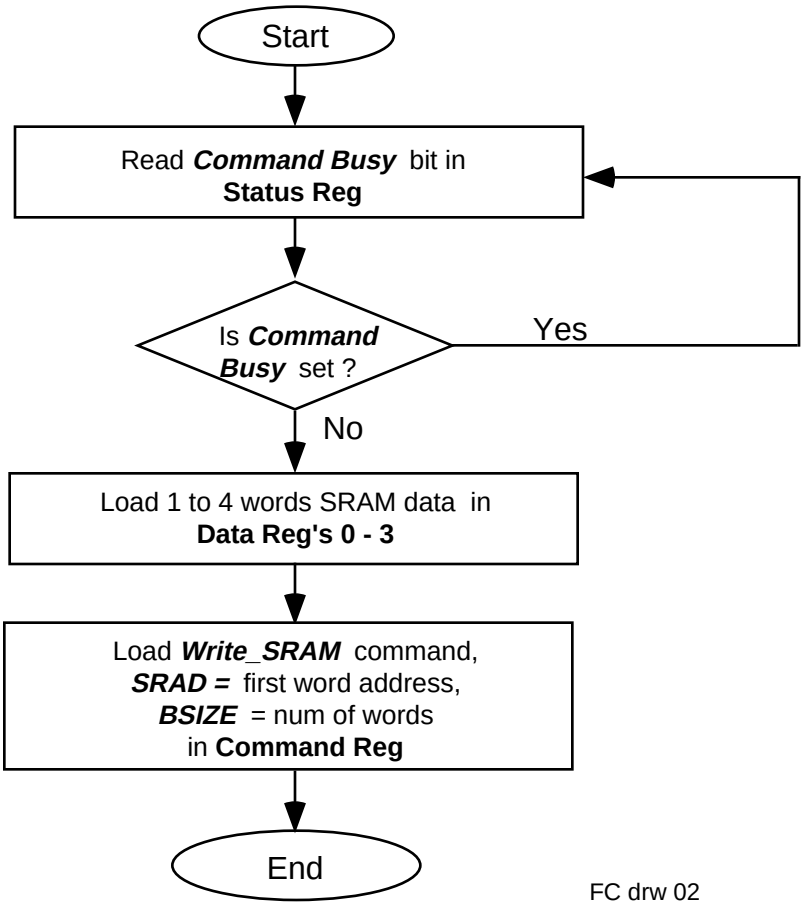
Note: Invalid Cell = GFC in ATM header !=0000b.

5 Recommended NICStAR Service Algorithm

This section provides example procedures to create and access different data structures associated with the NICStAR. The examples given in this section should be used as guidelines and should not be considered as the limitation of the NICStAR. Alternative ways can be used by studying the architecture of the NICStAR in this manual. Examples will be given in the forms of flow charts to describe the necessary steps in each example task. It is assumed that the reader of this section has already read the Theory of Operation and the Data Structure sections in this manual.

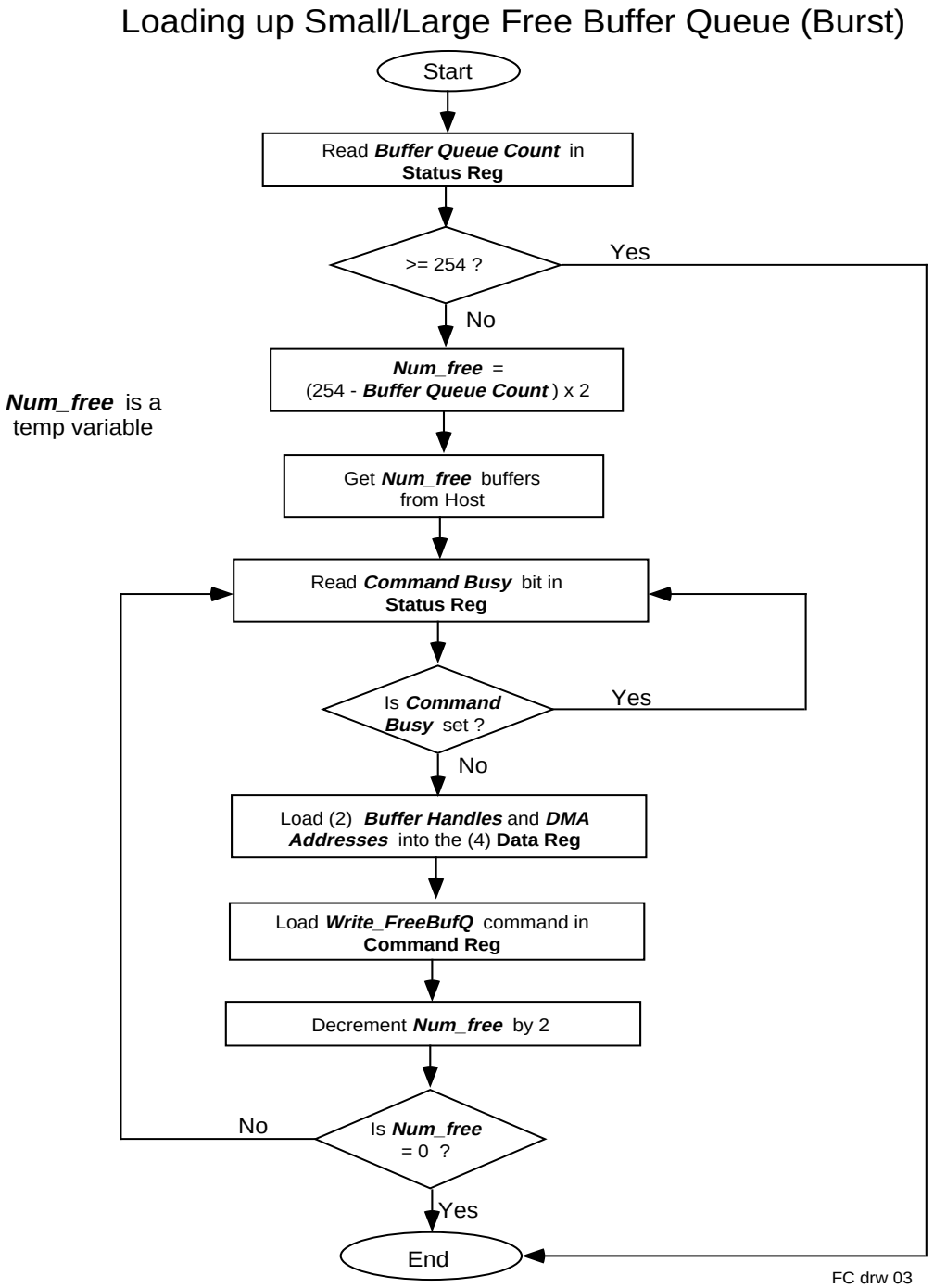

Reading Local SRAM
Start
Read **Command Busy** bit in **Status Reg**
Is **Command Busy** set ?
Yes
No
Load **Read_SRAM** command and **SRAD** = SRAM address in **Command Reg**
Read **Command Busy** bit in **Status Reg**
Is **Command Busy** set ?
Yes
No
Read **Data Reg 0** for SRAM data
End
FC drw 01

Writing Local SRAM

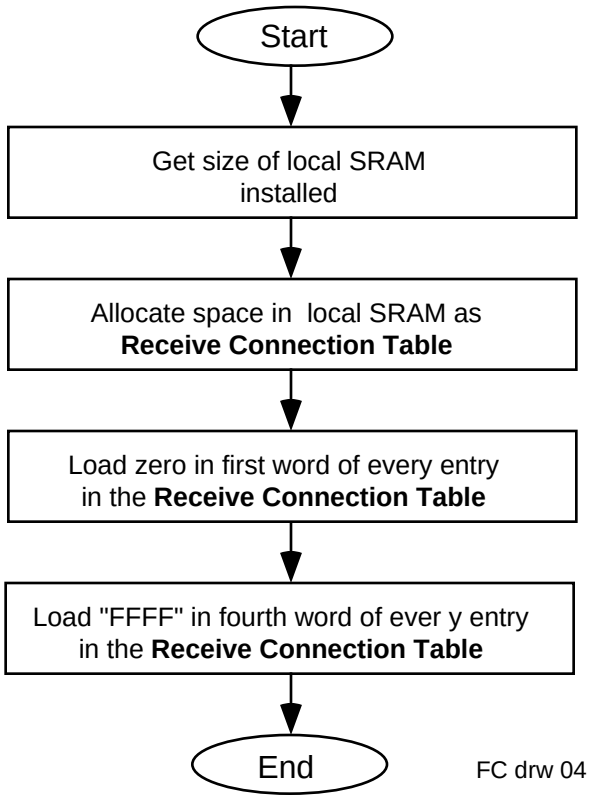


FC drw 02

5.3 Loading up Small/Large Free Buffer Queue (FBQ)



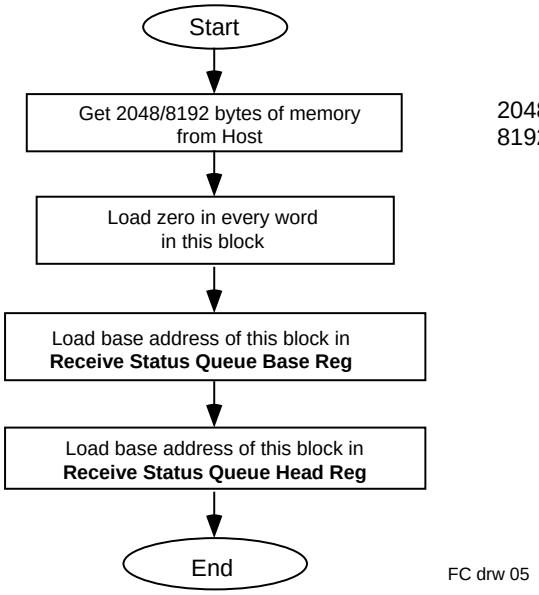
Receive Connection Table Initialization



FC drw 04

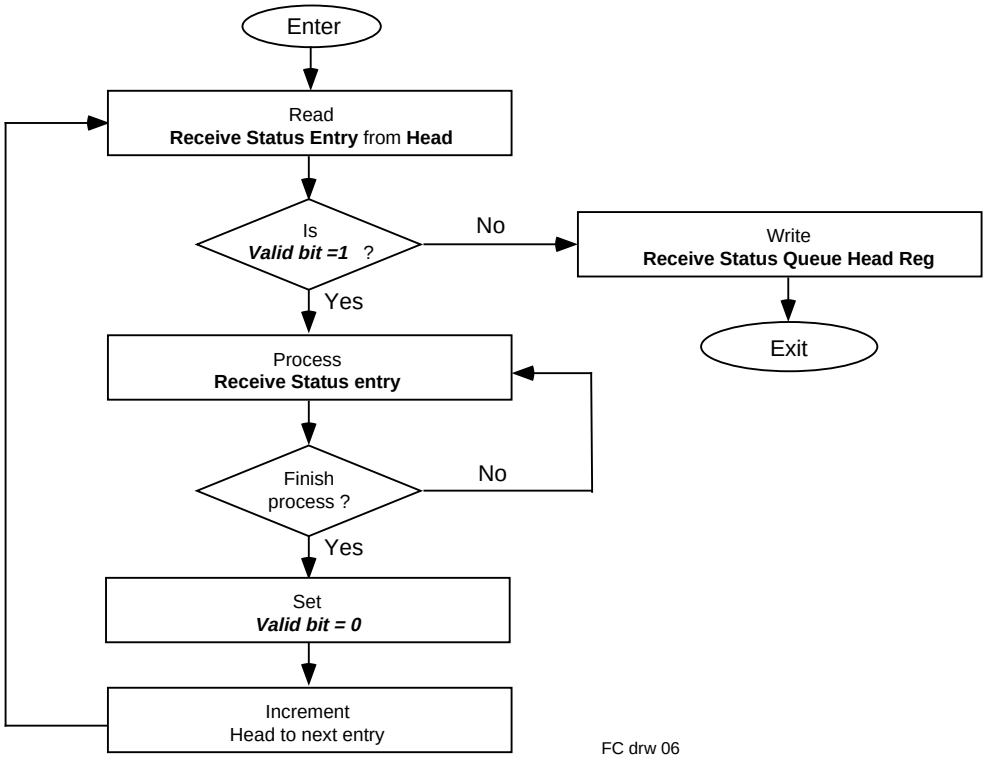
5.5 Allocate Receive Status Queue (RSQ) in Host Memory

Allocate Receive Status Queue (RSQ) in Host memory

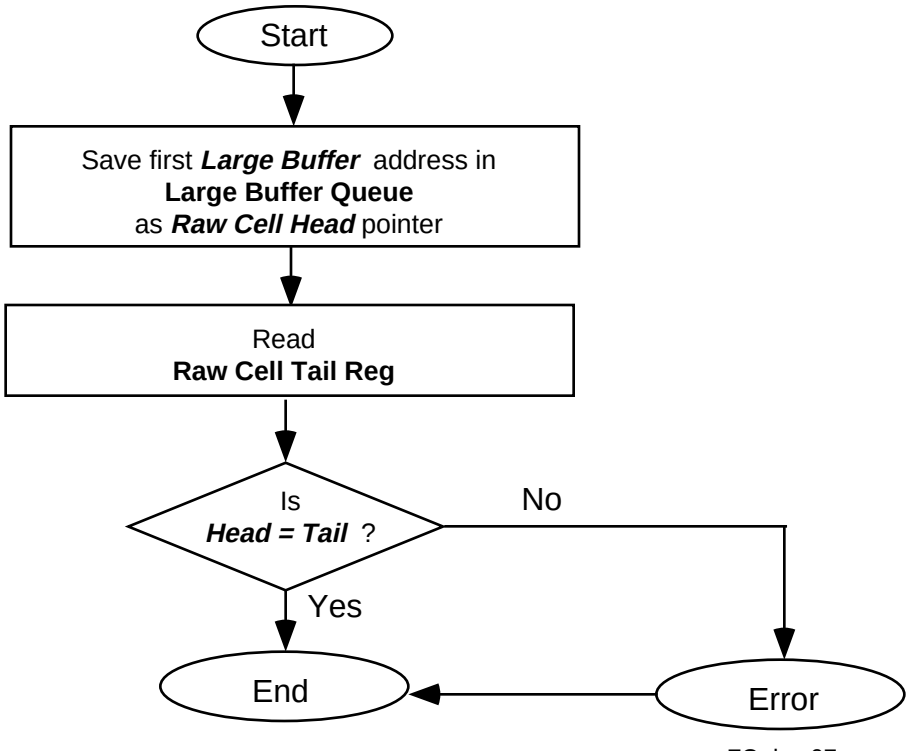


5.6 Service Receive Status Queue (RSQ)

Routine to De-queue (service) Receive Status Queue (RSQ)

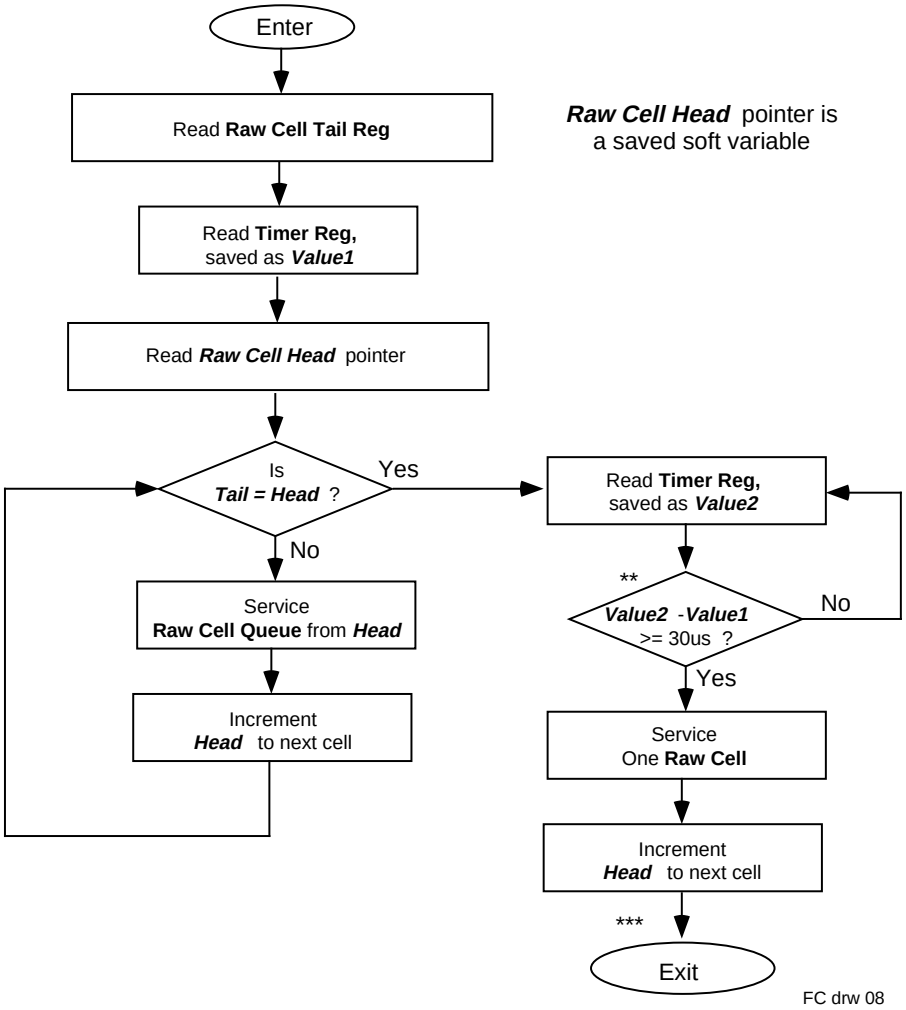


Raw Cell Queue Initialization



5.8 Service Raw Cell Queue (RCQ)

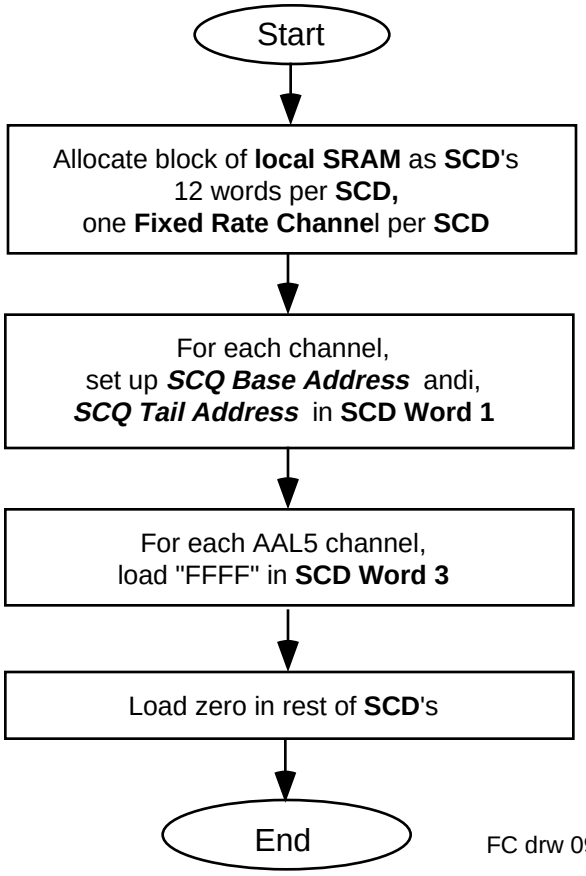
Service Raw Cell Queue



** The time parameter "30us" is the typical worst latency for the acquiring and transfer one cell (16 transfers in burst mode) across the PCI bus.
System operates with different worst case PCI latency should adjust this parameter accordingly.

*** For time critical Raw cell handling, device driver can process additional Raw cells and repeat this routine by checking the Tail pointer if it is advanced right before exit.

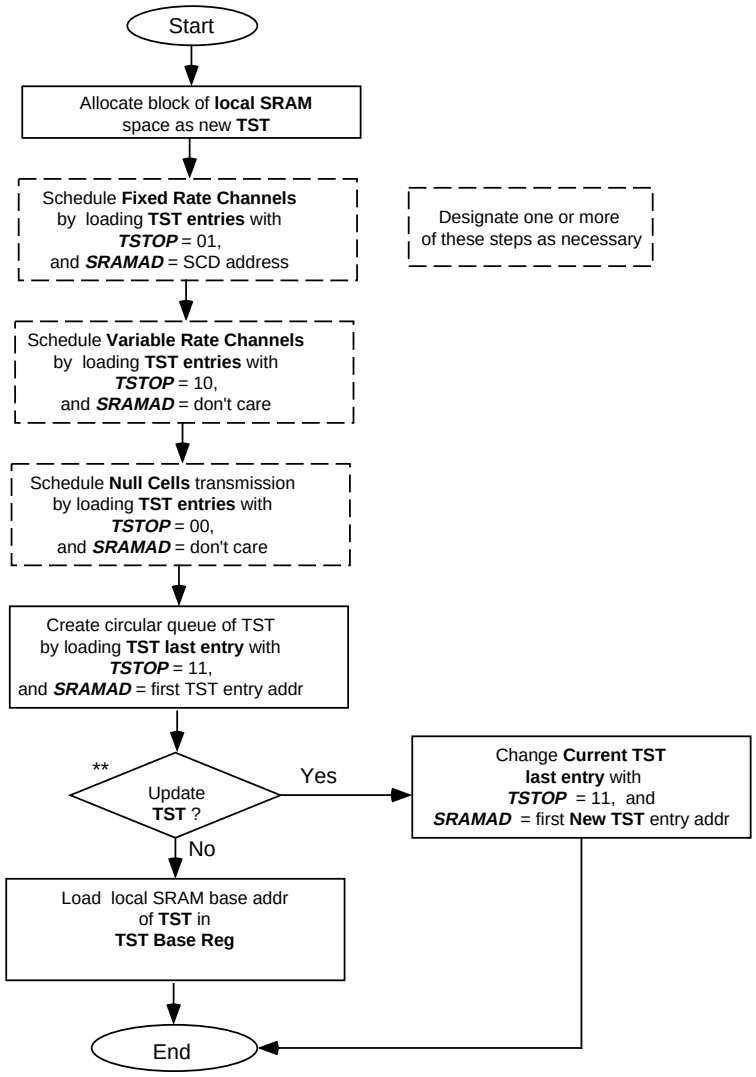
Create SCD for Fixed Rate Channels



FC drw 09

5.10 Create/Update Transmit Schedule Table (TST)

Create/update Transmit Schedule Table (TST) Example



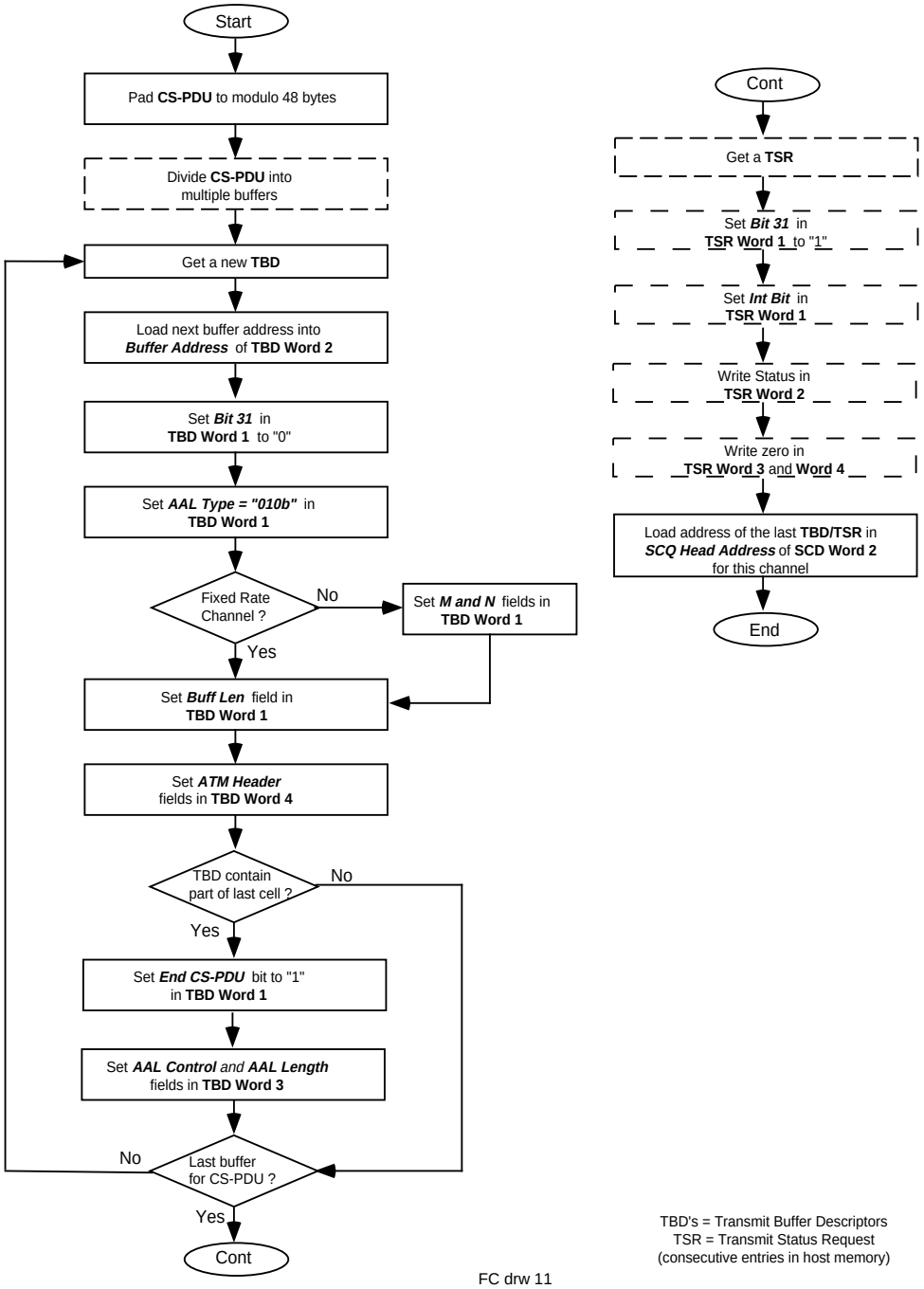
** In order to update **TST** on the fly without stopping transmit operation, two **TST**'s are used in this example.

TST is in local SRAM, see separate flow charts for reading and writing to local SRAM.

FC drw 10

5.11 Segmentation Routine for AAL5

Example Segmentation Routine for AAL5

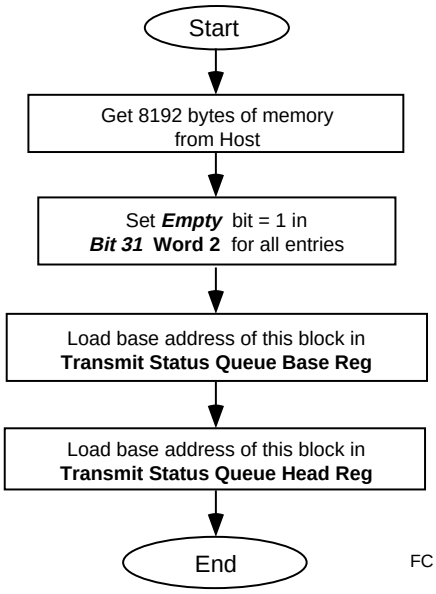


TBD's = Transmit Buffer Descriptors
TSR = Transmit Status Request
(consecutive entries in host memory)

FC drw 11

5.12 Create Transmit Status Queue (TSQ) in Host Memory

Create Transmit Status Queue (TSQ) in Host memory

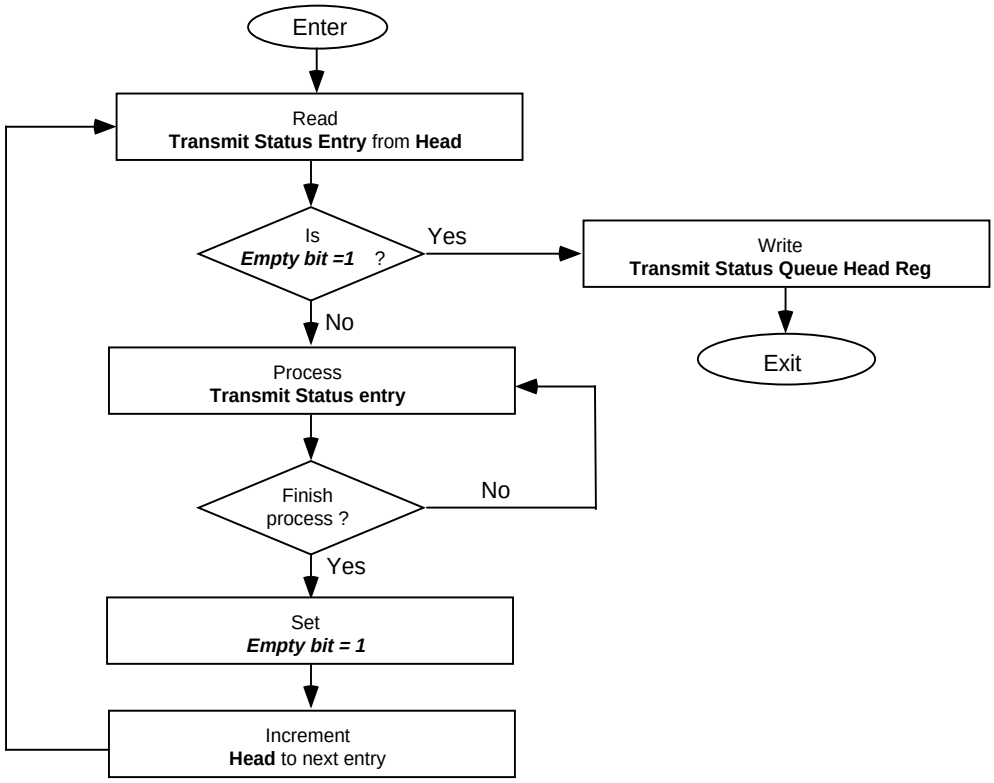


8192 bytes = 1024 entries;
(2 words per entry)

FC drw 12

5.13 Service Transmit Status Queue (TSQ)

Routine to De-queue (service) Transmit Status Queue (TSQ)



Head pointer is a saved soft variable

FC drw 13

6 Glossary of terms and acronyms

6.1 Terms and Acronyms used in this document

AAL	ATM Adaptation Layer.
CS-PDU	Convergence Sublayer Protocol Data Unit. The protocol data unit that NICStAR handles as a unit.
FBD	Free Buffer Descriptor, located in local SRAM An entry in the small or large FBQ. The FBD is a pointer to a receive free buffer.
FBQ	Free Buffer Queue, located in local SRAM The SAR maintains a list of small and large free buffers. These small and large FBQs are composed of FBDs.
Head Pointer	Pointer to a queue where the next entry will be read for processing.
PDU	Protocol Data Unit.
RCT	Receive Connection Table, located in local SRAM Table containing information on established received VCs - such as AAL type, buffer handle, cell count, etc.
RCQ	Raw Cell Queue, located in HOST memory A linked list of Large Free buffers used to store RM, OAM, unknown VPI/VCI and cells identified as "raw" in the RCT.
RSD	Receive Status Descriptor, located in HOST memory An entry in the RSQ that contains the buffer handle, CRC, VPI/VCI, etc. for a filled free buffer or completed CS-PDU.
RSQ	Receive Status Queue, located in HOST memory Contains an entry for each filled free buffer or completed CS-PDU. Entries are know as RSDs.
SAR	Segmentation and Reassembly.
SCD	Segmentation Channel Descriptor, located in local SRAM A pointer to a Segmentation Channel Queue (SCQ). There is one SCD for every SCQ.
SCQ	Segmentation Channel Queue, located in HOST memory A circular buffer containing TBDs and TSRs to be processed by the NICStAR. There is one SCQ for each fixed rate connection, and three (3) SCQs for all variable rate connections.

Tail Pointer	Pointer to a queue where the next entry will be written to or added to the queue.
TBD	Transmit Buffer Descriptor, located in HOST memory A pointer to data for the NICStAR to Transmit.
TSI	Transmit Status Indicator, located in HOST memory An entry in the TSQ that is generated when the NICStAR encounters a TSR in the SCQ. The TSI contains a status word and a time stamp.
TSQ	Transmit Status Queue, located in HOST memory A list of TSIs.
TSR	Transmit Status Request, located in HOST memory An entry in the SCQ which causes the NICStAR to generate a TSI in the TSQ. Allows the Device driver to monitor the NICStAR's progress/performance.
TST	Transmit Schedule Table, located in local SRAM A list containing nulls, jump instructions, and pointers to SCDs. The TST tells the NICStAR when and what to transmit. Each entry in the TST represents a cell transmission time.

6.2 ATM Acronyms

These acronyms are used through out ATM Forum literature and are included here for the users convenience when reading ATM documentation.

```
A - B

AAL  - ATM Adaptation Layer
ABR  - Available Bit Rate
ACM  - Address Complete Message
ACT  - Activity Bit
ADPCM  - Adaptive Differential Pulse Code Modulation

Ai - Signaling ID assigned by Exchange A
AIM - ATM Inverse Multiplexer
AIR - Additive Increase Rate
AIS - Alarm Indication Signal
AMI - Alternate Mark Inversion
ANI - Automatic Number Identification
ANM - Answer Message
ANSI - American National Standards Institute
API - Application Programming Interface
APPN - Advanced Peer to Peer Network
ARE - All Routes Explorer
ARP - Address Resolution Protocol
ARQ - Automated Repeat reQuest
ASE - Application Service Element
ASIC - Application Specific Integrated Circuit
ASN - Abstract Syntax Notation
ASP - Abstract Service Primitive
ATD - Asynchronous Time Division
ATM - Asynchronous Transfer Mode
ATMARP - ATM Address Resolution Protocol
ATS - Abstract Test Suite
AUU - ATM User-to-User
B-ICIB - Broadband Inter Carrier Interface
B-ICI SAAL - B-ICI signaling ATM Adaptation Layer
B-ISDN - Broadband Integrated Services Digital Network
B-ISUP - Broadband ISDN User's Part
B-LLI - Broadband Low Layer Information
B-NT - Broadband Network Termination
B-TE - Broadband Terminal Equipment
BBC - Broadband Bearer Capability
BCBDS - Broadband Connectionless Data Bearer Service
BCD - Binary Coded Decimal
BCOB - Broadband Class of Bearer
BECN - Backward Explicit Congestion Notification
BER - Bit Error Rate
BGP - Border Gateway Protocol
BGT - Broadcast and Group Translators
Bi - Signaling ID assigned by Exchange B
BIP - Bit Interleaved Parity
```

BISDN - Broadband - Integrated Services Digital Network
BISSI - Broadband Inter Switching System Interface
BN - Bridge Number
BOF - Birds of Feather
BOM - Beginning of Message
BOOTP - Bootstrap Protocol
BPDU - Bridge Protocol Data Unit
BPP - Bridge Port Pair
BPS - Bits per second
BSVC - Broadcast Switched Virtual Connections
BT - Burst Tolerance
BTAG - Begin Tag
BUS - Broadcast and Unknown Server
BW - Bandwidth

C - D

CA - Cell Arrival
CAC - Connection Admission Control
CBDS - Connectionless Broadband Data Service
CBR - Constant Bit Rate
CBR interactive - Constant Bit Rate interactive
CBR non-interactive - Constant Bit Rate non interactive
CC - Continuity Cell
CCITT - Consultative Committee on International Telephone &
Telegraph
CCR - Current Cell Rate
CCSS7 - Common Channel Signaling System 7
CDT - Cell Delay Tolerance
CDV - Cell Delay Variation
CDVT - Cell Delay Variation Tolerance
CEI - Connection Endpoint Identifier
CER - Cell Error Ratio
CES - Circuit Emulation Service
CI - Congestion Indicator
CIP - Carrier Identification Parameter
CIR - Committed Information Rate
CL - Connectionless
CLNAP - Connectionless Network Access Protocol
CLNP - Connectionless Network Protocol
CLNS - Connectionless Network Service
CLP - Cell Loss Priority
CLR - Cell Loss Ratio
CLS - Connectionless Server
CLSF - Connectionless Service Function
CME - Component Management Entity
CMI - Coded Mark Inversion
CMIP - Common Management Interface Protocol
CMR - Cell Misinsertion Rate
CN - Copy Network
CNM - Customer Network Management
CO - Connection Oriented
COD - Connection Oriented Data

COM - Continuation of Message
COS - Class of Service
CP - Connection Processor
CPCS - Common Part Convergence Sublayer
CPE - Customer Premises Equipment
CPG - Call Progress Message
CPI - Common Part Indicator
CPN - Customer Premises Network
CPN - Calling Party Number
Crankback IE - Crankback - Information Element
CRC - Cyclic Redundance Check
CRCG - Common Routing Connection Group
CRF(VC) - Virtual Channel Connection Related Function
CRF(VP) - Virtual Path Connection Related Function
CS - Convergence Sublayer
CS - Carrier Selection
CS1 - Capability Set One
CS2 - Capability Set Two
CSI - Convergence Sublayer Indication
CSPDN - Circuit Switched Public Data Network
CSR - Cell Missequenced Ratio
CSU - Channel Service Unit
CTD - Cell Transfer Delay
CTV - Cell Tolerance Variation
DA - Destination MAC address
DA - Destination Address
DCC - Data Country Code
DCE - Data Communication Equipment
DD - Depacketization Delay
DLC - Data Link Control
DES - Destination End System
DLCI - Data Link Connection Identifier
DMDD - Distributed Multiplexing Distributed Demultiplexing
DN - Distribution Network
DQDB - Distributed Queue Dual Bus
DS - Distributed Single Layer Test Method
DS-0 - Digital Signal, Level O
DS-1 - Digital Signal, Level 1
DS-2 - Digital Signal, Level 2
DS-3 - Digital Signal, Level 3
DS3 PLCP - Physical Layer Convergence Protocol
DSE - Distributed Single Layer Embedded Test Method
DSID - Destination Signaling Identifier
DSS2 - Setup Digital Subscriber Signaling #2
DSU - Data Service Unit
DTE - Data Terminal Equipment
DTL IE - DTL - Information Element
DXI - Data Exchange Interface

E - J

EFCI - Explicit Forward Congestion Indication
ELAN - Emulated Local Area Network

EMI - Electromagnetic Interference
EOM - End of Message
ETAG - End Tag
ETE - End-to-End
EXM - Exit Message
FC - Feedback Control
FCS - Fast Circuit Switching
FCS - Frame Check Sequence
FDDI - Fiber Distributed Data Interface
FEBE - Far End Block Error
FEC - Forward Error Correction
FERF - Far End Receive Failure
FRS - Frame Relay Service
FUNI - Frame User Network Interface
GAP - Generic Address Parameter
GCID - Global Call Identifier
GCID-IE - Global Call Identifier- Information Element
GCRA - Generic Cell Rate Algorithm
GFC - Generic Flow Control
GRC - Generic Reference Configuration
HDB3 - High Density Bipolar 3
HDLC - High Level Data Link Control
HEC - Header Error Control
HEL - Header Extension Length
HLPI - Higher Layer Protocol Identifier
HOL - Head of Line
IAA - Initial Address Acknowledgment
IAM - Initial Address Message
IAR - Initial Address Reject
IC - Initial Cell Rate
ICD - International Code Designator
ICMP - Internet Control Message Protocol
IDU - Interface Data Unit
IE - Information Element
IEC - Inter-exchange Carrier
IEEE - Institute of Electrical and Electronics Engineers
IETF - Internet Engineering Task Force
ILMI - Interim Link Management Interface
IOP - Interoperability
IP - Internet Protocol
Ipng - Internet Protocol Next Generation

IPX - Novell Internetwork Packet Exchange
ISO - International Organization for Standardization
ITU - International Telecommunications Union
IUT - Implementation Under Test
IWF - Interworking Function
IWU - Interworking Unit
JPEG - Joint Photographic Experts Group

L - O

LAN - Local Area Network

LANE - Local Area Network Emulation
LAPD - Link Access Procedure D
LB - Leaky Bucket
LD - LAN Destination
LE - LAN Emulation
LE_ARP - LAN Emulation Address Resolution Protocol
LEC - LAN Emulation Client
LEC - Local Exchange Carrier
LECID - LAN Emulation Client Identifier
LECS - LAN Emulation Configuration Server
LES - LAN Emulation Server
LIJP - Leaf Initiated Join Parameter
LIV - Link Integrity Verification
LLATMI - Lower Layer ATM Interface
LLC - Logical Link Control
LLC/SNAP - Logical Link Control/Subnetwork Access Protocol
LMI - Layer Management Interface
LOC - Loss of Cell delineation
LOF - Loss of Frame
LOS - Loss of Signal
LSB - Least Significant Bit
LSR - Leaf Setup Request
LT - Lower Tester
LTH - Length Field
MA - Maintenance and Adaptation
MAC - Medium Access Control
MAN - Metropolitan Area Network
MBS - Maximum Burst Size
MCR - Minimum Cell Rate
MCTD - Mean Cell Transfer Delay
ME - Mapping Entity
MIB - Management Information Base
MID - Message Identifier
MIN - Multistage Interconnection Networks
MIR - Maximum Information Rate
MMF - Multimode Fiberoptic cable
MPEG - Motion Picture Experts Group
MRCS - Multi-rate Circuit Switching
MS - Meta Signaling
MSAP - Management Service Access Point
MSB - Most Significant Bit
MSN - Monitoring Cell Sequence Number
MSVC - Meta-signaling Virtual Channel
MT - Message Type
MTP - Message Transfer Part
MTU - Message Transfer Unit
N-ISDN - Narrowband Integrated Services Digital Network
NDIS - Network Driver Interface Specification
NE - Network Element
NEBIOS - Network Basic Input/Output System
NHRP - Next Hop Resolution Protocol
NMS - Network Management System
NNI - Network to Network Interface

NP - Network Performance
NPC - Network Parameter Control
NRM - Network Resource Management
NSAP - Network Service Access Point
NSP - Network Service Provider
NSR - Non-Source Routed
NT - Network Termination
OAM - Operations, Administration and Maintenance
ODI - Open Data-Link Interface
OLI - Originating Line Information
OOF - Out of Frame
OPCR - Original Program Clock Reference
OSI - Open systems Interconnection
OSID - Origination Signaling Identifier
OSPF - Open Shortest Path First
OUI - Organization Unique Identifier

P - S

P-NNI - Private Network to Network Interface
PAD - Packet Assembler and Disassembler
PBX - Private Branch eXchange
PC - Priority Control
PCM - Pulse Code Modulation
PCO - Point of Control and Observation
PCR - Peak Cell Rate
PCR - Program Clock Reference
PCVS - Point to Point Switched Virtual Connections
PD - Packetization Delay
PDH - Plesiochronous Digital Hierarchy
PDU - Packet Data Unit
PHY - Physical Layer of the OSI model
PHY - Physical Layer
PICS - Protocol Implementation Conformance Statement
PID - Protocol Identifier Governing Connection Types
PIXIT - Protocol Implementation eXtra Information for Testing
PL - Physical Layer
PLL - Phase Locked Loop
PLPC - Physical Layer Convergence Protocol
PM - Physical Medium
PMD - Physical Layer Dependent sub-layer
POH - Path Overhead
POI - Path Overhead Indicator
PT - Payload Type
PTI - Payload Type Identifier
PVC - Permanent Virtual Circuit
PVCC - Permanent Virtual Channel Connection
PVPC - Permanent Virtual Path Connection
QD - Queuing Delay
QOS/QoS - Quality of Service
QPSX - Queue Packet and Synchronous Circuit Exchange
RAI - Remote Alarm Indication
RBOC - Regional Bell Operating Company

RC - Routing Control
RD - Route Descriptor
RDF - Rate Decrease Factor
RDI - Remote Defect Identification
REL - Release Message
RFC - Request For Comment (Document Series)
RFI - Radio Frequency Interference
RI - Routing Information
RII - Routing Information Indicator
RIP - Routing Information Protocol
RISC - Reduced Instruction Set Computing
RLC - Release Complete
RM - Resource Management
ROLC - Routing Over Large Clouds
RSVP (protocol) - Resource Reservation Protocol
RT - Routing Type
RTS - Residual Time Stamp
SA - Source MAC address
SA - Source Address
SAAL - Signaling ATM Adaptation Layer
SAP - Service Access Point
SAR - Segmentation and Reassembly
SCCP - Signaling Connection and Control Part
SCP - Service Control Point
SCR - Sustainable Cell Rate
SDH - Synchronous Digital Hierarchy
SDU - Service Data Unit
SE - Switching Element
SEAL - Simple and Efficient Adaptation Layer
SF - Switching Fabric
SGM - Segmentation Message
SID - Signaling Identifier
SIPP - SMDS Interface Protocol
SIR - Sustained Information Rate
SMC - Sleep Mode Connection
SMDS - Switched Multi-megabit Data Services
SMF - Single Mode Fiber
SN - Sequence Number
SNA - Systems Network Architecture
SNAP - Sub Network Access Protocol
SNMP - Simple Network Management Protocol
SOH - Section Overhead
SONET - Synchronous Optical Network
SPID - Service Protocol Identifier
SPTS - Single Program Transport Stream
SR - Source Routing (Bridging)
SRF - Specifically Routed Frame
SRT - Source Routing Transparent
SRTS - Synchronous Residual Time Stamp
SSCF - Service Specific Coordination Function
SSCOP - Service Specific Connection Oriented Protocol
SSCS - Service Specific Convergence Sublayer
ST - Segment Type

STE - Spanning Tree Explorer
STM - Synchronous Transfer Mode
STM1 - Synchronous Transport Mode 1 -- 155mbits/sec
STP - Signaling Transfer Point
STP - Shielded Twisted Pair cable
STS - Synchronous Time Stamps
STS-3c - Synchronous Transport System-Level 3 concatenated
SUT - System Under Test
SVC - Switched Virtual Circuit
SVCI - Switched Virtual Circuit Identifier
SVP - Switched Virtual Path
SWG - Sub-Working Group

T - Z

T1S1 - ANSI T1 Subcommittee
TB - Transparent Bridging
TC - Transaction Capabilities
TC - Transmission Convergence
TCAP - Transaction Capabilities Applications Part
TCI - Test Cell Input
TCO - Test Cell Output
TCP - Transmission Control Protocol
TCP - Test Coordination Procedure
TCP/IP - Transmission Control Program/Internet Protocol
TCS - Transmission Convergence Sublayer
TDJ - Transfer Delay Jitter
TDM - Time Division Multiplexing
TE - Terminal Equipment
TLV - Type / Length / Value
TM - Traffic Management
TM SWG - Traffic Management Sub-Working Group
TMP - Test Management Protocol
TNS - Transit Network Selection
TPCC - Third Party Call Control
TS - Traffic Shaping
TS - Time Stamp
TS - Transport Stream
TS - Time Slot
TSAP - Transport Service Access Point
UBR - Unspecified Bit Rate
UDP - User Datagram Protocol
UME - UNI Management Entity
UNI - User Network Interface
UPC - Usage Parameter Control
UT - Upper Tester
UTOPIA - Universal Test & Operations PHY Interface for ATM
UTP - Unshielded Twisted Pair cable
VBR - Variable Bit Rate
VBR delay sensitive - Variable Bit Rate delay sensitive
VBR delay tolerant - Variable Bit Rate delay tolerant
VBR non-interactive - Variable Bit Rate non-interactive
VC - Virtual Channel (Virtual Circuit)

VC-Multiplexing - Virtual Channel - Multiplexing
VCC - Virtual Channel Connections
VCI - Virtual Circuit Identifier
VCI - Virtual Connection Identifier
VCI - Virtual Channel Identifier
VLAN - Virtual Local Area Network
VP - Virtual Path
VP/VC - Virtual Path, Virtual Circuit
VPC - Virtual Path Connection
VPCI/VCI - Virtual Path Connection Identifier/Virtual Channel Identifier
VPI - Virtual Path Identifier
VS - Virtual Scheduling
WAN - Wide Area Network
XNS - Xerox Network Systems
XTP - eXpress Transport Protocol

7 Appendix

7.1 Appendix A - Jump start programming example

The purpose of this "coding" example is to provided as simple example to get the user on the correct path to transmitting and receiving cells using NICStAR. It demonstrates a set up to transmit on two channels and receive on two channels using the AAL5 format.

The code is is written in a pseudo assembly code format where the following operations are used:

"PCI IO Write"	-	Read or Write in I/O address space
"Write to SRAM"	-	Write to location in NICStAR local SRAM using command and data registers of NICStAR
"Write to Host"	-	Write to host local DRAM. Target address specifies 32-bit physical address.
"Load 2 small Free Buffer"	-	Load two free buffers into the small free buffer queue in the NICStAR local SRAM using the load free buffer command and the data registers.

```
/----------------------------  Start File -------------------------------------/
```

Date:10-13-95

Example NICStAR Configuration for Transmitting two fixed channels and Receiving two VCs. using AAL5 PDU's.

The following setup assumes that the PCI Bios has configured the SAR PCI registers at power-up/reset.

```
//----------------------- System Configuration -------------------------
// Reset NICStAR with S/W.

  Operation        Address       Data           Comments
/----------\   /---------------\ /---------\  /---------------------------------------\
PCI IO Write   SarIoBase + 0x14, 0x80000000     Set SAR Reset Bit for Min. 2 PCI clocks.
PCI IO Write   SarIoBase + 0x14, 0x00000000     Set SAR Out of Reset Bit


/-------------------------------------------------------------------------------\
|
|               Configure NICStAR for Transmit
|
\-------------------------------------------------------------------------------/

// Setup first fixed channel Tx SCD

  Operation         Address        Data                        Comments
/-----------\  /------------\    /----------\ /---------------------------------------\
Write to Sram  0x1C000           0x00000000     First word of SCD address  = 0x4000
                                                This address was picked because it was
                                                convenient.
Write to Sram  0x1C001           0x00000000     Second word of SCD address = 0x4001
Write to Sram  0x1C002           0xFFFFFFFF     Third word of SCD address  = 0x4002
                                                Init. AAL5 CRC

// Setup Second fixed channel Tx SCD


  Operation        Address         Data                        Comments
/-----------\  /-----------\     /-----------\ /---------------------------------\
Write to Sram  0x1C00C           0x00000000     First word of SCD address  = 0x400C
                                                This address was picked to allow 8 words
                                                for first fixed channel TBD cache A and
                                                 cache B.
Write to Sram  0x1C00D           0x00000000     Second word of SCD address = 0x400D
Write to Sram  0x1C00E           0xFFFFFFFF     Third word of SCD address  = 0x400E.
                                                Init. AAL5 CRC
```

```
//---------------------- Setup Tx Schedule Table ----------------------

 Operation       Address         Data                 Comments
/-----------\ /----------\ /------------\ /-----------------------------------------\
Write to Sram  0x1C100       0x20004000    Service SCD 1 at address 0x4000. 33.3% BW.
Write to Sram  0x1C101       0x2000400C    Service SCD 2 at address 0x400C  33.3% BW.
Write to Sram  0x1C102       0x00000000    Force a NULL Cell to Tx.   Optional. 33.3% BW
Write to Sram  0x1C103       0x60004100    Start (JUMP) at the beginning of TST table
                                           address 4100.

// Setup TST Base Register

  Operation          Address          Data                   Comments
/----------\   /--------------\ /--------\  /-----------------------------------------\
PCI IO Write   SarIoBase +0x3C,  0x00010400     The TST address 0x4100 is shifted left by
                                               two bits. When the NICStAR Tx is enabled,
                                               the first TST table gets read at address
                                               0x4100.
PCI IO Write   SarIoBase +0x40,  TxStatusQBase Tx Status Queue Base Address.
                                               Bits 12:0
                                               must be 0x0.




//------------------------- Setup Host Memory --------------------------------

// Setup Host Tx Buffer Desc. TBD for fixed channel 1.


  Operation      Address         Data                  Comments
/---------\     /-------------\   /-----------\     /---------------------------------\
Write to Host   CH1TBDQBase +0x0    0x480000C0      One AAL5 PDU, length = 4 cells.
Write to Host   CH1TBDQBase +0x4    HTxBuf1         Address of Tx buffer 1.
Write to Host   CH1TBDQBase +0x8    0x000000B5      AAL5 Control = 0x0000,
                                                    AAL5 Length = 0xB5.

Write to Host   CH1TBDQBase+0xC   0x07512340        Cell Header.
                                                    GFC = 0x0, VPI = 0x75,
                                                    VCI = 0x1234, PTI = 0x0, CLP = 0x0.
                                                    To do a cable loop-back change to
                                                    VPI = 0x00 VCI=0x0002
```

// Setup Host Tx Buffer Desc. TBD for fixed channel 2.

Operation	Address	Data	Comments
Write to Host	CH2TBDQBase +0x0	0x480001B0	One AAL5 PDU, length = 9 cells.
Write to Host	CH2TBDQBase +0x4	HTxBuf2	Address of Tx buffer 2.
Write to Host	CH2TBDQBase +0x8	0x000001A0	AAL5 Control = 0x0000, AAL5 Length = 0x1A0
Write to Host	CH2TBDQBase +0xC	0x04932590	Cell Header. GFC = 0x0, VPI = 0x49, VCI = 0x3259, PTI = 0x0, CLP = 0x0. To do a cable loop-back change to VPI = 0x00, VCI = 0x000

// Enable Tx.

Operation	Address	Data	Comments
PCI IO Write	SarIoBase +0x14,	0x00000020	Enable Tx. The TST table is scanned. Channel 1 and Channel 2 SCD's are tested if Head and Tails are equal.

//--------- At this point the NICStAR is ready to segment the two channels --------

// After the following operation the NICStAR Segments and Transmits channel 1.

Operation	Address	Data	Comments
Write to Sram	0x04000	CH1TBDQBase + 0x10	The Tail pointer is moved to point to the next TBD in the Queue. i.e. making the Head and Tail pointers different.
Write to Sram	0x0400C	CH2TBDQBase + 0x10	The Tail pointer is moved to point to the next TBD in the Queue. i.e. making the Head and Tail pointers different

// ------ At this point the SAR start segmenting both fixed rate channels. --------

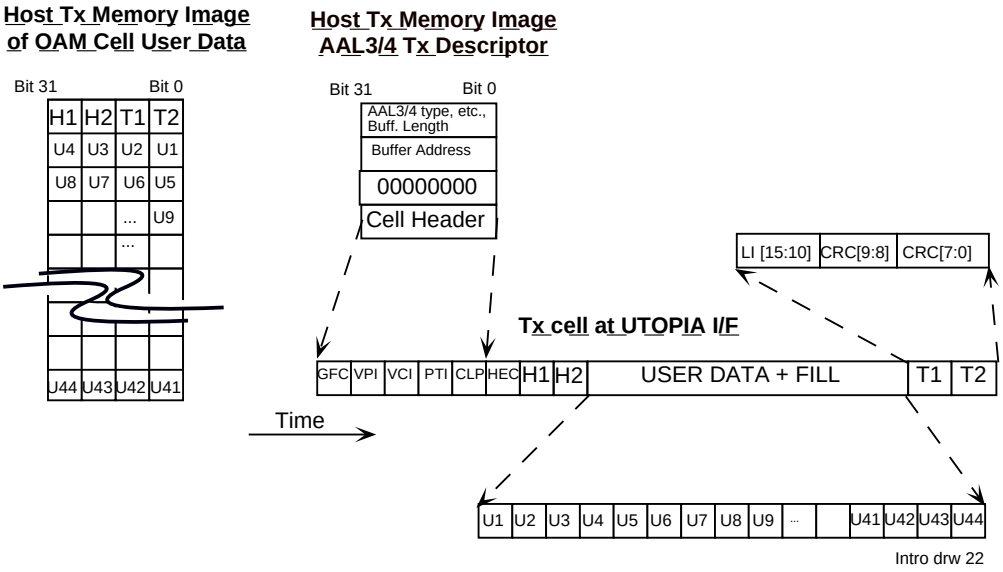
```
/-------------------------------------------------------------------------------\
|
|                              Configure NICStAR for Receive
|
\-------------------------------------------------------------------------------/
// Setup first fixed channel Rx Connection Table. VPI = 0x0, VCI = 0x0001, AAL5.
Operation         Address  Data              Comments
/-----------\  /------\    /---------\       /----------------------------------\
Write to Sram  0x00004     0x000A0000        Open Connection, AAL5.
Write to Sram  0x00005     0x00000000        Rx Buffer Handle.
Write to Sram  0x00006     0x00000000        Host Rx DMA Address.
Write to Sram  0x00007     0xFFFFFFFF        Rx. AAL5 Partial CRC. Init. to FFFFFFFF

// Setup Second Rx Connection Table. VPI = 0x0, VCI = 0x0002, AAL5.
Operation      Address     Data              Comments
/-----------\  /-----\     /---------\       /----------------------------------\
Write to Sram  0x00008     0x000A0000        Open Connection, AAL5.
Write to Sram  0x00009     0x00000000        Rx Buffer Handle.
Write to Sram  0x0000A     0x00000000        Host Rx DMA Address.
Write to Sram  0x0000B     0xFFFFFFFF        Rx. AAL5 Partial CRC. Init. to FFFFFFFF

Operation                  Host Address         Comments
/------------------------\ /--------------\  /----------------------------------\
Load 2 small Free Buffer   HSFB1, HSFB2      Host Small Free Buffer 1 and 2.
Load 2 small Free Buffer   HSFB3, HSFB4      Host Small Free Buffer 3 and 4.
Load 2 small Free Buffer   HSFB5, HSFB6      Host Small Free Buffer 5 and 6.
Load 2 small Free Buffer   HSFB7, HSFB8      Host Small Free Buffer 7 and 8.
Load 2 large Free Buffer   HLFB1, HLFB2      Host Small Free Buffer 1 and 2.
Load 2 large Free Buffer   HLFB3, HLFB4      Host Small Free Buffer 3 and 4.
Load 2 large Free Buffer   HLFB5, HLFB6      Host Small Free Buffer 5 and 6.
Load 2 large Free Buffer   HLFB7, HLFB8      Host Small Free Buffer 7 and 8.


// For loading small and large free buffers see users manual.
Operation        Address           Data           Comments
/-----------\  /---------------\ /----------\   /------------------------------\
PCI IO Write   SarIoBase +0x1C   RSQB           Receive Status Queue Base Address.
PCI IO Write   SarIoBase +0x14,  0x20000020     Enable Receive and Transmit.

// At this point any Rx cell is received in the Rx (315 cell) FIFO then gets filtered in
the Rx Connection Table.  Cells belonging to the VCI = 0x2 and 0x3 gets re-assembled in
the receive free buffers.
```

```
// Address variables.
//-------------------------------------------------------------------------------\
|
|TxStatusQBase    Tx Status Queue Base Address.32 bit word.    Bits 12:0 must be 0x0.
|CH1TBDQBase      Channel 1 Tx Buffer Descriptor Queue Base.   Bits  9:0   "  "   "
|CH2TBDQBase      Channel 2 Tx Buffer Descriptor Queue Base.   Bits  9:0   "  "   "
|HTxBuf1          Host Tx Buffer 1 start address. Any byte alignment.
|                 The channel 1 cell data contents are stored in this data buffer.
|
|HTxBuf2          Host Tx Buffer 2 start address. Any byte alignment.
|                 The channel 2 cell data contents are stored in this data buffer.
|
|RSQB             Receive Status Queue Base Address. Bits 10:0 must be set to 0x0.
|
|
|HSFB1                        Host Small Free Buffer 1 Address. Must be word aligned.
|HSFB2                        Host Small Free Buffer 2 Address.        "       "
|HSFB3                        Host Small Free Buffer 3 Address.        "       "
|HSFB4                        Host Small Free Buffer 4 Address.        "       "
|HSFB5                        Host Small Free Buffer 5 Address.        "       "
|HSFB6                        Host Small Free Buffer 6 Address.        "       "
|HLFB1                        Host Large Free Buffer 1 Address.        "       "
|HLFB2                        Host Large Free Buffer 2 Address.        "       "
|HLFB3                        Host Large Free Buffer 3 Address.        "       "
|HLFB4                        Host Large Free Buffer 4 Address.        "       "
|HLFB5                        Host Large Free Buffer 5 Address.        "       "
|HLFB6                        Host Large Free Buffer 6 Address.        "       "
|
|
\-------------------------------------------------------------------------------/
```

7.2 *Appendix B - Generate OAM Cell using AAL3/4 Channel*

This application note is to illustrate how to generate (transmit) ATM OAM cells by using the NICStAR AAL3/4 channel feature. The advantage of this approach is that CRC-10 of the ATM cell is generated by the NICStAR hardware. On receiving, the OAM cells are received as Raw Cells and they will go to the Raw Cell Queue.

Transmit OAM cells using AAL3/4 channel:

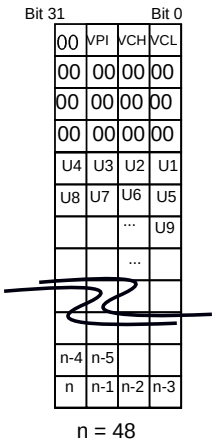


Note: H1, H2 and LI[15:10] can be any value placed in the host memory as needed by the OAM cell.
CRC[9:0] is generated by the NICStAR hardware at the UTOPIA I/F.

Receive OAM cells as Raw Cells:

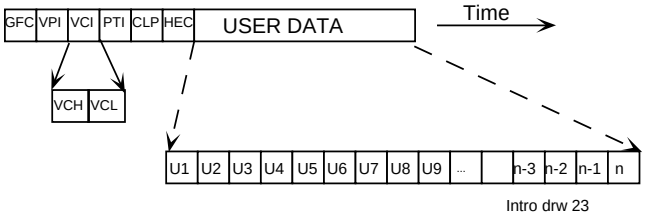
Rx RAW CELL/OAM

Host Rx Memory Image of Raw Cell



n = 48

RAW Cell at UTOPIA I/F

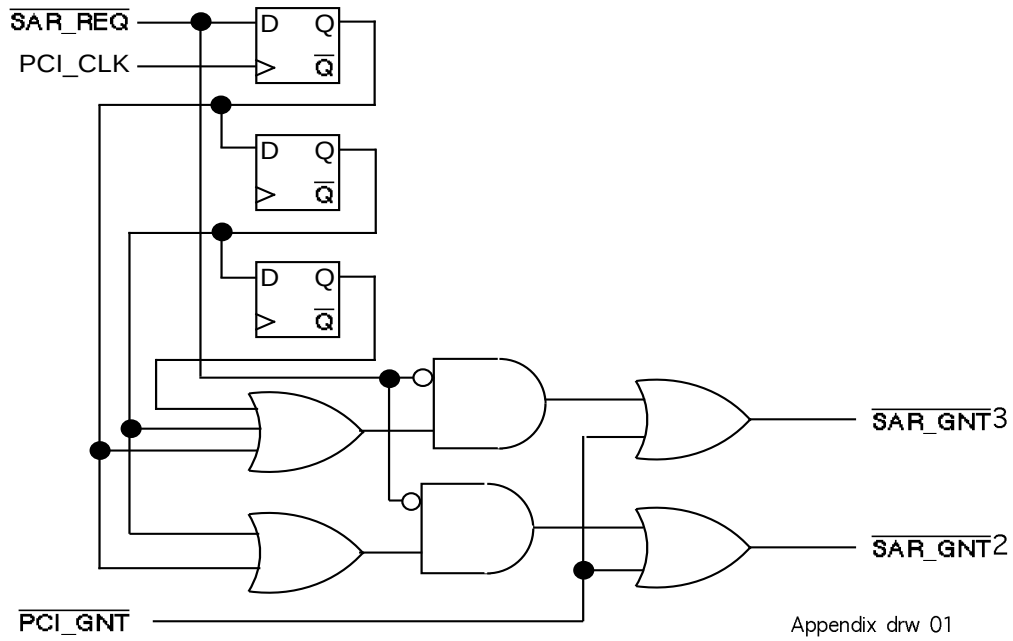


Intro drw 23

Note: VCH = most significant 8 bits of the VCI value,
VCL = least significant 8 bits of the VCI value.

7.3 Appendix C - PAL diagram and Equation for MOD 1 Operation of the 77211

This is the diagram and equation for the PAL used to enable the 77211 to run in a MOD 1 (any byte boundry) mode. It is not required unless MOD 1 is desired




```
MODULE revefix

TITLE 'SAR PCI Grant bug fix pal - M.Mueller Copyright 1996 IDT,Inc.'

                    PAL Solution for the IDT77211 (Mod 1 issue)

        ********************************************************

                              PROGRAM DESCRIPTION

        ********************************************************

"REVISION HISTORY
" REV    DATE           DESCRIPTION
" -----    --------   -------------------------------------
"   00    9/26/96          NEW PROGRAM
"*******************************************************

                              DECLARATIONS

********** INPUTS **********
pci_clk         pin 1; "PCI CLK
pci_rst_        pin 2; "RST# from PCI
pci_gnt_         pin 3; "GNT# from PCI
sar_req_         pin 4; "REQ# from SAR
utl_ad0         pin 5; "Utility bus AD(0)
```

```
utl_cs0_       pin 6; "Utility bus CS(0)#
utl_cs1_       pin 7; "Utility bus CS(1)#
nc1            pin 8; "
nc2            pin 9; "
pal_oe_        pin 11;"PAL output enable for reg outputs - tied to ground

"***************  OUTPUTS

blk_gnt        pin 12;
blkgnt_r       pin 13;
blkgnt_r2      pin 14;
nc9            pin 15;
nc10           pin 16;
sar_gnt2_      pin 17; "GNT# to SAR 155Mbit
sar_gnt3_      pin 18; "GNT# to SAR 25Mbit
req_en_        pin 19; "bus switch enable for SAR_REQ# to REQ#

"*********************************************************

EQUATIONS

blk_gnt.clk = pci_clk;
blkgnt_r.clk = pci_clk;
blkgnt_r2.clk = pci_clk;

"req_en_ is used to control the bus switch which blocks REQ#
"from the SAR. With Rev E this is not needed, so the bus switch is
"set up to be closed (enabling the SAR).

req_en_ = 0;

blk_gnt := sar_req_;
blkgnt_r := blk_gnt.fb;
blkgnt_r2 := blkgnt_r.fb;

"Toggle sar_gnt_ when the bus is parked with the SAR, and then
"the SAR requests the bus (the converse to the rev C3/D bug).

sar_gnt2_ =  pci_gnt_
        # ((blk_gnt.fb # blkgnt_r.fb) & !sar_req_);

sar_gnt3_ =  pci_gnt_
        # ((blk_gnt.fb # blkgnt_r.fb # blkgnt_r2.fb) & !sar_req_);

"*********************************************************
"                END OF PROGRAM
"*********************************************************
```

TEST_VECTORS
([pci_clk,pal_oe_, pci_gnt_, sar_req_, pci_rst_] ->
[blk_gnt,blkgnt_r,blkgnt_r2, sar_gnt2_,sar_gnt3_, req_en_])

[.C.,0, 1, 1, 0] -> [1,.x.,.x.,1,1, 0];"INITIALIZE
[.C.,0, 1, 1, 0] -> [1,1,.x.,1,1, 0];"INITIALIZE
[.C.,0, 1, 1, 0] -> [1,1,1, 1,1, 0];"INITIALIZE
[0 ,0, 0, 0, 0] -> [1,1,1, 1,1, 0];"Check flow through of pci_gnt_ to
[0 ,0, 0, 1, 0] -> [1,1,1, 0,0, 0];"sar_gnt_
[0 ,0, 1, 0, 1] -> [1,1,1, 1,1, 0];
[0 ,0, 1, 1, 1] -> [1,1,1, 1,1, 0];
[1 ,0, 1, 1, 1] -> [1,1,1, 1,1, 0];
[1 ,0, 1, 0, 1] -> [1,1,1, 1,1, 0];
[1 ,0, 0, 1, 1] -> [1,1,1, 0,0, 0];
[1 ,0, 0, 0, 1] -> [1,1,1, 1,1, 0];
[0 ,0, 0, 0, 1] -> [1,1,1, 1,1, 0];
[1 ,0, 0, 0, 1] -> [0,1,1, 1,1, 0];
[1 ,0, 0, 1, 1] -> [0,1,1, 0,0, 0];
[1, 0, 1, 0, 1] -> [0,1,1, 1,1, 0];
[1, 0, 1, 1, 1] -> [0,1,1, 1,1, 0];
[0, 0, 1, 1, 1] -> [0,1,1, 1,1, 0];
[0, 0, 1, 0, 1] -> [0,1,1, 1,1, 0];
[0, 0, 0, 0, 1] -> [0,1,1, 1,1, 0];
[0, 0, 0, 1, 1] -> [0,1,1, 0,0, 0];
[1, 0, 0, 1, 1] -> [1,0,1, 0,0, 0];
[1, 0, 1, 1, 1] -> [1,0,1, 1,1, 0];
[0, 0, 1, 1, 1] -> [1,0,1, 1,1, 0];
[1, 0, 1, 1, 1] -> [1,1,0, 1,1, 0];
[0, 0, 1, 1, 1] -> [1,1,0, 1,1, 0];
[1, 0, 1, 1, 1] -> [1,1,1, 1,1, 0];
[0, 0, 1, 1, 1] -> [1,1,1, 1,1, 0];
[1, 0, 1, 1, 1] -> [1,1,1, 1,1, 0];
[0, 0, 1, 1, 1] -> [1,1,1, 1,1, 0];
[1, 0, 1, 1, 1] -> [1,1,1, 1,1, 0];
[0 ,0, 0, 0, 0] -> [1,1,1, 1,1, 0];
[0 ,0, 0, 1, 0] -> [1,1,1, 0,0, 0];
[0 ,0, 1, 0, 1] -> [1,1,1, 1,1, 0];
[0 ,0, 1, 1, 1] -> [1,1,1, 1,1, 0];
[1 ,0, 1, 1, 1] -> [1,1,1, 1,1, 0];
[1 ,0, 1, 0, 1] -> [1,1,1, 1,1, 0];
[1 ,0, 0, 1, 1] -> [1,1,1, 0,0, 0];
[1 ,0, 0, 0, 1] -> [1,1,1, 1,1, 0];
[0 ,0, 0, 0, 1] -> [1,1,1, 1,1, 0];
[1 ,0, 0, 0, 1] -> [0,1,1, 1,1, 0];
[1 ,0, 0, 1, 1] -> [0,1,1, 0,0, 0];
[1, 0, 1, 0, 1] -> [0,1,1, 1,1, 0];
[1, 0, 1, 1, 1] -> [0,1,1, 1,1, 0];
[0, 0, 1, 1, 1] -> [0,1,1, 1,1, 0];
[0, 0, 1, 0, 1] -> [0,1,1, 1,1, 0];

```
[ 0, 0, 0, 0, 1] -> [0,1,1, 1,1, 0];
[ 0, 0, 0, 1, 1] -> [0,1,1, 0,0, 0];
[ 1, 0, 0, 1, 1] -> [1,0,1, 0,0, 0];
[ 1, 0, 1, 1, 1] -> [1,0,1, 1,1, 0];
[ 0, 0, 1, 1, 1] -> [1,0,1, 1,1, 0];
[ 1, 0, 1, 1, 1] -> [1,1,0, 1,1, 0];
[ 0, 0, 1, 1, 1] -> [1,1,0, 1,1, 0];
[ 1, 0, 1, 1, 1] -> [1,1,1, 1,1, 0];
[ 0, 0, 1, 1, 1] -> [1,1,1, 1,1, 0];
[ 1, 0, 1, 1, 1] -> [1,1,1, 1,1, 0];

[ 0, 0, 0, 1, 1] -> [1,1,1, 0,0, 0];
[ 1, 0, 0, 1, 1] -> [1,1,1, 0,0, 0];
[ 0, 0, 0, 0, 1] -> [1,1,1, 1,1, 0];
[ 1, 0, 0, 0, 1] -> [0,1,1, 1,1, 0];"clk1
[ 0, 0, 0, 0, 1] -> [0,1,1, 1,1, 0];
[ 1, 0, 0, 0, 1] -> [0,0,1, 0,1, 0];"clk2
[ 0, 0, 0, 0, 1] -> [0,0,1, 0,1, 0];
[ 1, 0, 0, 0, 1] -> [0,0,0, 0,0, 0];"clk3
[ 0, 0, 0, 1, 1] -> [0,0,0, 0,0, 0];
[ 1, 0, 0, 1, 1] -> [1,0,0, 0,0, 0];
[ 0, 0, 0, 1, 1] -> [1,0,0, 0,0, 0];
[ 1, 0, 0, 1, 1] -> [1,1,0, 0,0, 0];
[ 0, 0, 0, 1, 1] -> [1,1,0, 0,0, 0];
[ 1, 0, 0, 1, 1] -> [1,1,1, 0,0, 0];

END
```

INDEX

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W